НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації та управління

"На правах рукопису"
УДК *004.089*

До захисту допущено
В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“ ____ ” _____ 20 *20* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за освітньо-професійною програмою

«*Інформаційні управляючі системи та технології*»

зі спеціальності 126 «*Інформаційні системи та технології*»

на тему:

«Методи та засоби семантичного аналізу текстів»

Виконав:
студент VI курсу, групи **ІС-92мп**
Мигаль Дмитро Степанович _____

Керівник:
старший викладач,
Олійник Юрій Олександрович _____

Консультант:
професор, д.т.н., доцент,
Жаріков Едуард В’ячеславович _____

Рецензент:
доц. каф. ТК, к.т.н., доцент,
Ткач М.М _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації та управління

Рівень вищої освіти – *другий (магістерський)*

Спеціальність – 126 «*Інформаційні системи та технології*»

Освітньо-професійна програма *«Інформаційні управляючі системи та технології»*

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

«__» _____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Мигаль Дмитро Степанович

1. Тема дисертації «Методи та засоби семантичного аналізу текстів», науковий керівник дисертації Олійник Юрій Олександрович, старший викладач, затверджені наказом по університету від «26» жовтня 2020 р. № 3132-с

2. Строк подання студентом дисертації “ 2 ” 12 20 20 р.

3. Об’єкт дослідження математичне, інформаційне та програмне забезпечення людино-машинного спілкування українською мовою

4. Перелік завдань, які потрібно розробити
Аналіз сучасних методів та засобів семантичного аналізу текстів; огляд існуючих ресурсів обробки та аналізу україномовних текстів; розробка методу LSA з підтримкою обробки україномовних текстів; огляд та вибір технологій для реалізації методів та засобів; розробка програмного забезпечення; дослідження ефективності розробленого методу та програмного забезпечення.

5. Орієнтовний перелік графічного (ілюстративного) матеріалу

Діаграма потоків даних у системі
Діаграма діяльності
Креслення екранних форм
Відображення результатів семантичного дослідження слова «тисяча»
Структура розробленого програмного застосунку
Ефективність роботи системи на базі розробленої моделі семантичного аналізу
Графік залежності загальної кількості слів від кількості документів у корпусі

6. Орієнтовний перелік публікацій

Дві публікації: одні тези доповіді на міжнародній науково-практичної конференції «TOPICAL ISSUES OF THE DEVELOPMENT OF MODERN SCIENCE», одні тези доповіді на науково-практичній конференції «Інформатика та обчислювальна техніка-ІОТ-2020»

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання “ *1* ” *вересня* 20 *20* р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Аналіз сучасних методів та засобів семантичного аналізу текстів*	*11.09*	
2	*Огляд існуючих ресурсів обробки та аналізу україномовних текстів*	*20.09*	
3	*Розробка методу LSA з підтримкою обробки україномовних текстів*	*3.10*	
4	*Огляд та вибір технологій для реалізації методів та засобів*	*16.10*	
5	*Розробка програмного забезпечення*	*31.10*	
6	*Дослідження ефективності розробленого методу та програмного забезпечення*	*4.11*	
7	*Оформлення документації*	*15.11*	
8	*Подання роботи на попередній захист*	*20.11*	
9	*Подання роботи на основний захист*	*02.12*	

Студент _____ *Дмитро МИГАЛЬ*

Науковий керівник _____ *Юрій ОЛІЙНИК*

РЕФЕРАТ

Магістерська дисертація: 91 с., 39 рис., 8 табл., 22 джерела, 1 додаток.

Актуальність. У подальшому майбутньому, створення нових методів семантичного аналізу текстів відкриє нові можливості та дозволить істотно просунутися у вирішенні багатьох завдань прикладної лінгвістики, таких як машинний переклад, автореферування, класифікація текстів і т.п. Не менш актуальною є і розробка нових засобів та інструментів, що дозволяють автоматизувати семантичний аналіз. Подібні методи аналізу дозволяють збирати основну інформацію про певну тематику, спрямованість і настрій текстів, що в подальшому буде спрощувати автоматизовану роботу з ними, таку як каталогізація, пошук і порівняння. Застосування семантичних моделей є актуальним в автоматизованих навчальних системах, при вирішенні певних задач, таких як: вилучення знань з текстів, інформаційного пошуку, реферування, контролю коректності словникових термінів і визначень, автоматичної генерації асоціативних зв’язків в гіпертекстових базах даних тощо.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Методи та технології високопродуктивних обчислень та обробки надвеликих масивів даних». Державний реєстраційний номер 0117U000924.

Мета роботи – підвищення якості семантичного аналізу україномовних текстів за рахунок вдосконалення семантичних моделей та моделей граматики української мови з урахуванням предметної області тексту.

Для досягнення мети необхідно виконати наступні **завдання:**

- аналіз сучасних методів та засобів семантичного аналізу текстів;
- огляд наявних ресурсів обробки та аналізу україномовних текстів;
- розробка методу LSA з підтримкою обробки україномовних текстів;
- огляд та обґрунтування вибору технологій для реалізації методів та засобів;

- розробка програмного забезпечення;
- дослідження ефективності розробленого методу та програмного забезпечення.

Об'єктом дослідження роботи є математичне, інформаційне та програмне забезпечення людино-машинного спілкування українською мовою.

Предметом дослідження є моделі і методи семантичного аналізу україномовного тексту.

Методи дослідження, застосовані у даній роботі, базуються на методах семантичного аналізу текстів.

Наукова новизна – вдосконалені методи семантичного аналізу, що містять підтримку обробки україномовних текстів.

Прикладна значущість. Методи та програмне забезпечення може бути використане для семантичного аналізу даних україномовних текстів, що підтримано проектом концепції розвитку штучного інтелекту України.

Публікації. Матеріали роботи опубліковані в тезах міжнародної науково-практичної конференції «TOPICAL ISSUES OF THE DEVELOPMENT OF MODERN SCIENCE» та у тезах науково-практичної конференції «Інформатика та обчислювальна техніка-IOT-2020».

СЕМАНТИЧНИЙ АНАЛІЗ, ТЕМАТИЧНЕ МОДЕЛЮВАННЯ, УКРАЇНСЬКА МОВА, LATENT SEMANTIC ANALYSIS, NATURAL LANGUAGE PROCESSING

ABSTRACT

Master's dissertation: 91 pp., 39 figs., 8 tables, 14 sources, 1 appendix.

Topicality. The creation of new methods of semantic analysis of texts will open new opportunities and allow us to significantly progress in solving many problems of computational linguistics, such as machine translation, authoring, text classification, etc. No less important is the development of new tools and instruments to automate semantic analysis. Such analysis methods allow us to collect basic information about the subject, focus, and mood of the texts, further simplifying the automated work with them, such as cataloging, search, and comparison. The use of semantic models is relevant in automated learning systems, extracting knowledge from texts, information retrieval, abstracting, checking the correctness of dictionaries of terms and definitions, automatic generation of associative links in hypertext databases, and more.

Connection of work with scientific programs, plans, themes. The work was performed at the Department of Automated Information Processing and Control Systems of the National Technical University of Ukraine "Kyiv Polytechnic Institute. Igor Sikorsky ”within the theme“ Methods and means of semantic analysis of texts ”.

The purpose of the work - improving the quality of semantic analysis of Ukrainian-language texts by improving the semantic models and models of grammar of the Ukrainian language, taking into account the subject area of the text.

To achieve this goal you must perform the following **tasks**:

- analysis of modern methods and means of semantic texts analysis;
- review of existing resources for processing and analysis of Ukrainian-language texts;
- development of the LSA method with support for processing Ukrainian-language texts;
- review and selection of technologies for the implementation of methods and tools;
- software development;
- study of the effectiveness of the developed method and software.

The object of research is mathematical, informational and software of human-

machine communication in the Ukrainian language.

The subject of the research are models and methods of semantic analysis of the Ukrainian text.

The research methods used in this paper are based on the methods of semantic analysis of texts.

Scientific novelty - improved methods of semantic analysis, which include support for processing Ukrainian-language texts.

Applied significance. Methods and software can be used for semantic analysis of data of Ukrainian-language texts, which is supported by the draft concept of development of artificial intelligence of Ukraine.

Publications. The materials of the work are published in the abstracts of the international scientific-practical conference "TOPICAL ISSUES OF THE DEVELOPMENT OF MODERN SCIENCE" and in the abstracts of the scientific-practical conference "Informatics and Computer Engineering-IOT-2020".

SEMANTIC ANALYSIS, THEMATIC MODELING, UKRAINIAN LANGUAGE, LATENT SEMANTIC ANALYSIS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 9
ВСТУП 10
1 АНАЛІЗ СУЧАСНИХ ЗАСОБІВ СЕМАНТИЧНОГО АНАЛІЗУ ТЕКСТІВ . 13
1.1 Засоби семантичного аналізу текстів іноземними мовами 13
1.1.1 Засіб виявлення відносин та закономірностей в англомовних текстах InfraNodus 13
1.1.2 Система контекстного та порівняльного подання англомовного тексту WordWanderer 15
1.1.3 Система візуалізації та пошуку інформації в текстових документах WordTree 17
1.1.4 Система семантичного аналізу російськомовних текстів Istio 19
1.2 Засоби семантичного аналізу україномовних текстів 21
1.3 Методи тематичного моделювання природно-мовних текстів 23
1.3.1 Векторна модель текстів 23
1.3.2 Латентно-семантичний аналіз 26
1.3.3 Імовірнісний летентно-семантичний аналіз 28
1.3.4 Латентне розміщення Діріхле 29
Висновки до розділу 31
2 УДОСКОНАЛЕНИЙ МЕТОД СЕМАНТИЧНОГО АНАЛІЗУ ТЕКСТОВИХ ДАНИХ 33
2.1 Розроблення удосконаленого методу семантичного аналізу україномовних текстів 33
2.2 Приклад застосування розробленого методу 34
Висновки до розділу 41
3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ 42
3.1 Використані технології 42
3.1.1 Технологія Node.js 42
3.1.2 Мова Python 43
3.1.3 Формат даних JSON 44

3.2 Керівництво користувача 45
3.3 Опис системи 54
Висновки до розділу 64
4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ 65
4.1 Опис ідеї стартап-проекту 65
4.2 Аналіз ринкових можливостей стартап-проекту 66
4.3 Розроблення маркетингової стратегії для стартап продукту 71
Висновки до розділу 73
5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО МЕТОДУ 74
5.1 Аналіз швидкодії моделі 74
5.2 Розподілений режим обробки 76
Висновки до розділу 77
ВИСНОВКИ 78
ПЕРЕЛІК ПОСИЛАНЬ 80
ДОДАТОК А Графічний матеріал 83
Діаграма потоків даних у системі 84
Діаграма діяльності 85
Креслення екранних форм 86
Відображення результатів семантичного дослідження слова «тисяча» 87
Структура розробленого програмного застосунку 88
Ефективність роботи системи на базі розробленої моделі семантичного аналізу 89
Графік залежності загальної кількості слів від кількості документів У корпусі 90

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VSM Vector Space Model, з англійської векторна модель текстів.

LSA Latent Semantic Analysis, з англійської латентно-семантичний аналіз.

pLSA Probabilistic Latent Semantic Analysis, з англійської імовірнісний латентно-семантичний аналіз.

LDA Latent Dirichlet Allocation, з англійської латентне розміщення Діріхле.

SVD Singular Value Decomposition, з англійської сингулярний розклад матриці.

API Application Programming Interface, з англійської прикладний програмний інтерфейс.

B2C Business-to-consumer, з англійської бізнес для споживача.

B2B Business-to-business, з англійської бізнес для бізнесу.

MVP Minimum viable product, з англійської мінімально життєздатний продукт.

ВСТУП

Автоматична обробка текстових документів набуває все більшої необхідності та значущості у сучасному світі. Попит на аналіз великої кількості документів зростає щодня, адже це зумовлено накопиченням великої кількості текстової інформації в мережі за рахунок мільйонів веб-сайтів та застосунків. За статистикою, розмір World Wide Web, станом на травень 2020 року, оцінюється близько 5.47 мільярдів сторінок [1]. Хоча, оскільки це базується на індексованих сторінках з великих пошукових системах, таких як Google, Bing та Yahoo, то можна вважати, що обсяг документів у всесвітній мережі інтернет значно перевищує цю цифру. Тому, очевидним є те, що без допомоги автоматичної обробки, аналізувати інформацію такого об’єму неможливо.

За оцінкою топ десяти мільйонів веб-сторінок, україномовний контент містить близько 0.3%, що в десятки разів менше ніж російськомовний – 8.6% та значно менший ніж відсоток англомовного контенту – 59.6% [2]. Саме тому, на даний час не існує багато програмних рішень автоматичного семантичного аналізу української мови, порівняно з англійською та російською.

На даний момент у світі існує кілька методів знаходження сенсу висловлювань, проте жоден з них не є універсальним. Подібні методи аналізу дозволяють збирати основну інформацію про тематику, спрямованості і настрої текстів, що в подальшому спрощує автоматизовану роботу з ними, таку як каталогізація, пошук і порівняння. Найбільш популярним напрямом вилучення інформації з текстів на даний момент є використання різних статистичних методів для обробки тексту, наприклад, побудова частотних словників, конкорданс, порівняння з використанням виділених сутностей і тощо.

Створення нових методів семантичного аналізу текстів відкриє нові можливості та дозволить істотно просунутися у вирішенні багатьох завдань комп'ютерної лінгвістики, таких як машинний переклад, автореферування, класифікація текстів і т.п. Не менш актуальною є розробка нових засобів та інструментів, що дозволяють автоматизувати семантичний аналіз. Подібні

методи аналізу дозволяють збирати основну інформацію про тематику, спрямованість і настрій текстів, що в подальшому спрощує автоматизовану роботу з ними, таку як каталогізація, пошук і порівняння. Застосування семантичних моделей є актуальним в автоматизованих навчальних системах, при вирішенні задач вилучення знань з текстів, інформаційного пошуку, реферування, контролю коректності словників термінів і визначень, автоматичної генерації асоціативних зв'язків в гіпертекстових базах даних тощо.

На даний час успішно вирішена задача морфологічного аналізу текстів, результати якого застосовуються в пошукових Інтернет-машинах, текстових редакторах, підсистемах перевірки орфографії тощо. Найбільш популярним напрямом вилучення інформації з текстів на даний момент є використання різних статистичних методів для обробки тексту, наприклад, побудова частотних словників, конкорданс (словників словосполучень), порівняння з використанням виділених сутностей і тощо.

Семантика - розділ лінгвістики, що вивчає смислове значення одиниць мови. Вона задається певною математичною моделлю, яка описує певні обчислення, які є можливими у мові. Семантичний аналіз дає точне або словникове значення зі структур, створених синтаксичним аналізом. Основна мета семантичного аналізу - це мінімізувати структури синтаксису та знайти їх значення завдяки пошуку синонімів, розбору сенсу слова, перекладу на інші мови, а також заповнення бази знань. Завдання семантичного аналізу, по своїй суті, не вирішено в повній мірі. Синтаксичний аналіз можна зустріти в системах перекладу і у підсистемах перевірки граматики. Незважаючи на доволі багату теорію в області семантичного аналізу, застосування у сучасних сферах робочих середовищ знаходять лише методи аналізу, які були засновані на статистичних характеристиках слів і словосполучень аналізованого тексту. Слід зазначити - підсистеми, що реалізують зазначені методи аналізу тексту, не надають методів налаштування процесу аналізу, а також засобів поповнення баз правил граматики мови.

Семантичні моделі тексту, що є результатом комплексного аналізу, дозволяють оцінити коректність тексту в наочній формі, візуально уявити структуру сюжету, взаємозв'язок об'єктів і процесів тексту та їх атрибути. Послідовність моделей простих речень тексту і результуюча візуальна модель тексту дозволяють реалізувати зворотний зв'язок "вплив на модель - реакція в тексті", завдяки чому в повній мірі можна в інтерактивному режимі налагоджувати процеси аналізу текстів та бачити докази об'єктивності та однозначності тлумачення текстів на природних мовах.

1 АНАЛІЗ СУЧАСНИХ ЗАСОБІВ СЕМАНТИЧНОГО АНАЛІЗУ ТЕКСТІВ

1.1 Засоби семантичного аналізу текстів іноземними мовами

Наразі існує доволі багато засобів семантичного аналізу, які в загальному випадку відштовхуються та напряму залежать від поставленої задачі та мови. В даному підрозділі розглянуто лише засоби аналізу мов, які є близькі за своєю будовою до української, а саме російську та англійську. Вони є найбільш поширеними у сучасному світі за своєю чисельністю використання у мережі, а саме статтях, письмових роботах та текстах.

1.1.1 Засіб виявлення відносин та закономірностей в англомовних текстах InfraNodus

У наш час, усе більшої популярності набирають сервіси-інструменти мережевого мислення, які застосовуються у робочих процесах а також у світовій мережі інтернет. Одним з таких сервісів є InfraNodus - інструмент мережевого мислення, який виявляє відносини та закономірності в корпусі текстів. Вхідними даними можуть бути: звичайний набір текстів, PDF-файли, електроні таблиці, Twitter, Google, Evernote чи RSS-канали тощо. InfraNodus цікавий тим, що реалізує метод аналізу текстової мережі, що базується на побудові тексту у вигляді певного графу, який виводить результат на екран з можливістю для дослідника побачити його наочно. Вузли у ньому представляють слова, а ребра - їхні співпадіння у текстах. Після представлення корпусу текстів таким чином застосовується широкий спектр інструментів мережевого аналізу для виявлення кластерів, тісно пов'язаних між собою термів (тематичне моделювання), виявлення найвпливовіших вузлів (ключових слів), виконання кількісного та якісного аналізу текстів, оцінку структури, визначення структурних прогалин та порівняльного аналізу текстів. Більше того, візуальне подання тексту як мережі допомагає забезпечити більш послідовне вираження ідей, чіткіше показати та прослідкувати певний родинно-

наслідковий зв’язок між словами та їх значеннями, саме тому інструмент може знайти корисне застосування у писемній творчості, лінгвістичних дослідженнях, та в освітньому контексті. Подібні інструменти, наприклад iVisClustering, TopicNets та VisiRR, є недоступними для широкого загалу, їх вихідний код закритий, вони в основному орієнтовані на вивчення великого корпусу текстів, а їх застосування обмежується моделюванням тем та пошуком тексту на основі ключових слів. InfraNodus доступний як в Інтернеті, так і як окрема версія програми з відкритим кодом. Його інтерфейс спеціально розроблений для обробки тексту в режимі реального часу, забезпечуючи візуалізацію та аналіз корпусу в режимі реального часу із можливістю додавання нових даних. Крім того, InfraNodus пропонує аналіз структури дискурсу, що забезпечує вимірювання рівня упередженості та виявлення певних структурних прогалин [3].

Підхід, реалізований в InfraNodus, вдосконалює існуючі методи пошуку змісту в тексті, що використовують такі методи, як прихований семантичний аналіз або LSA, pLSA, розподіл Пачинко, приховане виділення Діріхле або LDA, моделі реляційних тем, алгоритм word2vec, та його розширення lda2vec. Ці методи засновані на отриманні центральних тем з тексту шляхом ідентифікації кластерів спільних слів, які поєднуються між собою векторами-зв’язками і перетворюють загальний цілісний текст на павутину слів та прямих зв’язків між ними. Потім ці дані можуть бути використані для класифікації подібних документів, вдосконалення методів індексації та пошуку тексту а також виявлення розвитку певних тем протягом певного періоду в межах певного корпусу тексту [3]. Більш наглядний приклад програми, а також її вигляд інтерфейсу можна побачити на рисунку 1.1.

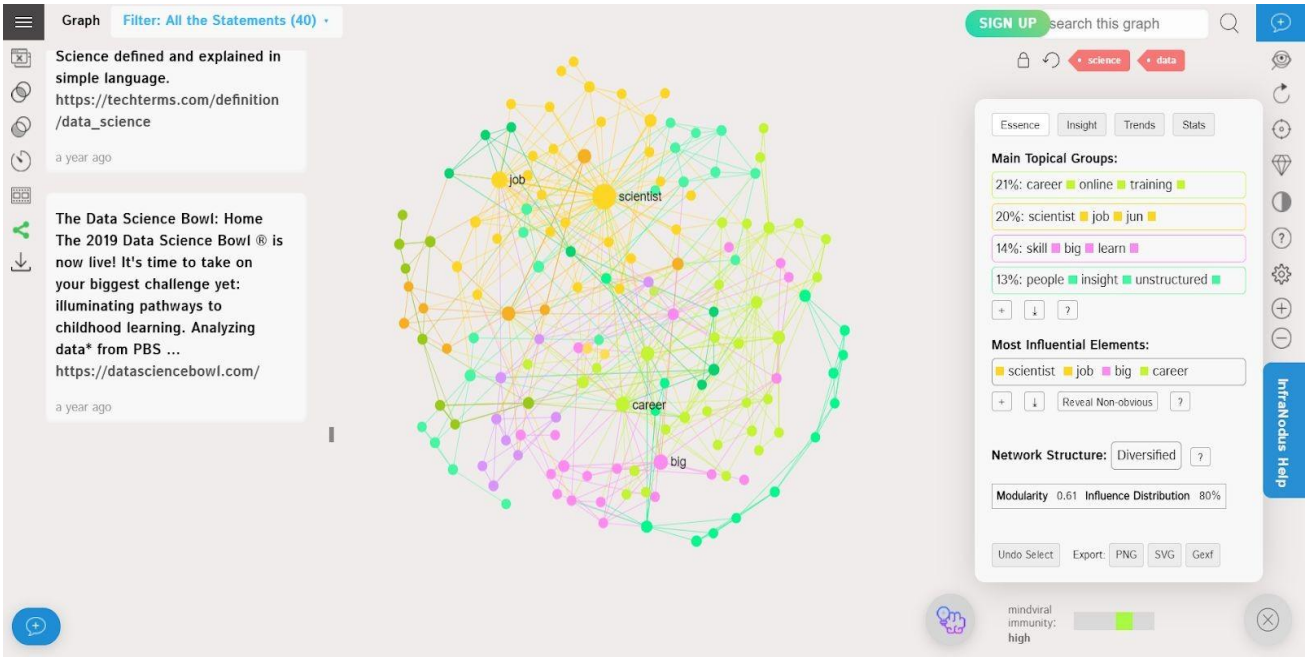


Рисунок 1.1 - Інтерфейс веб-сервісу InfraNodus

1.1.2 Система контекстного та порівняльного подання англомовного тексту WordWanderer

Іще однією популярною системою у наш час є система WordWanderer (рисунок 1.2). У ній реалізована техніка візуалізації, яка збільшує можливості звичайної хмари тегів до навігаційного інтерфейсу для тексту. Даний інструмент підтримує функціонал від "контекстного подання", що представляє усі слова, які зустрічаються поруч із обраним словом, до "порівняльного подання", що розташовує слова на основі міри їх асоціації а також значенням та смисловим зв’язком між собою. Значною частиною системи WordWanderer є візуалізації, яка може мати один із трьох станів: хмара (слово не обрано), контекст (одне слово виділене) та подання порівняння (вибрано два слова). Враховуючи, що розмір шрифту було визначено як одну з особливо корисних візуальних змінних для проектування хмар тегів, система використовує його для кодування загальної частоти у вигляді хмари та міри асоціації в контексті та порівняння. Вертикальне розташування усіх заданих у системі слів є послідовно алфавітним у всіх поданнях, тоді як горизонтальне розташування змінюється залежно від режиму.

У хмарному поданні відображаються найпоширеніші слова тексту з використанням розміру шрифту для представлення відносних частот. Порядок слів в алфавітному порядку зверху вниз та зліва направо. При наведенні курсору на слово можна помітити, як виділяється кольором його зв’язок з іншими словами в залежності від міри їхнього співвідношення в тексті. Всі кольори можна задати самостійно, єдиним обмеженням є кількість наявних кольорів у програмі.

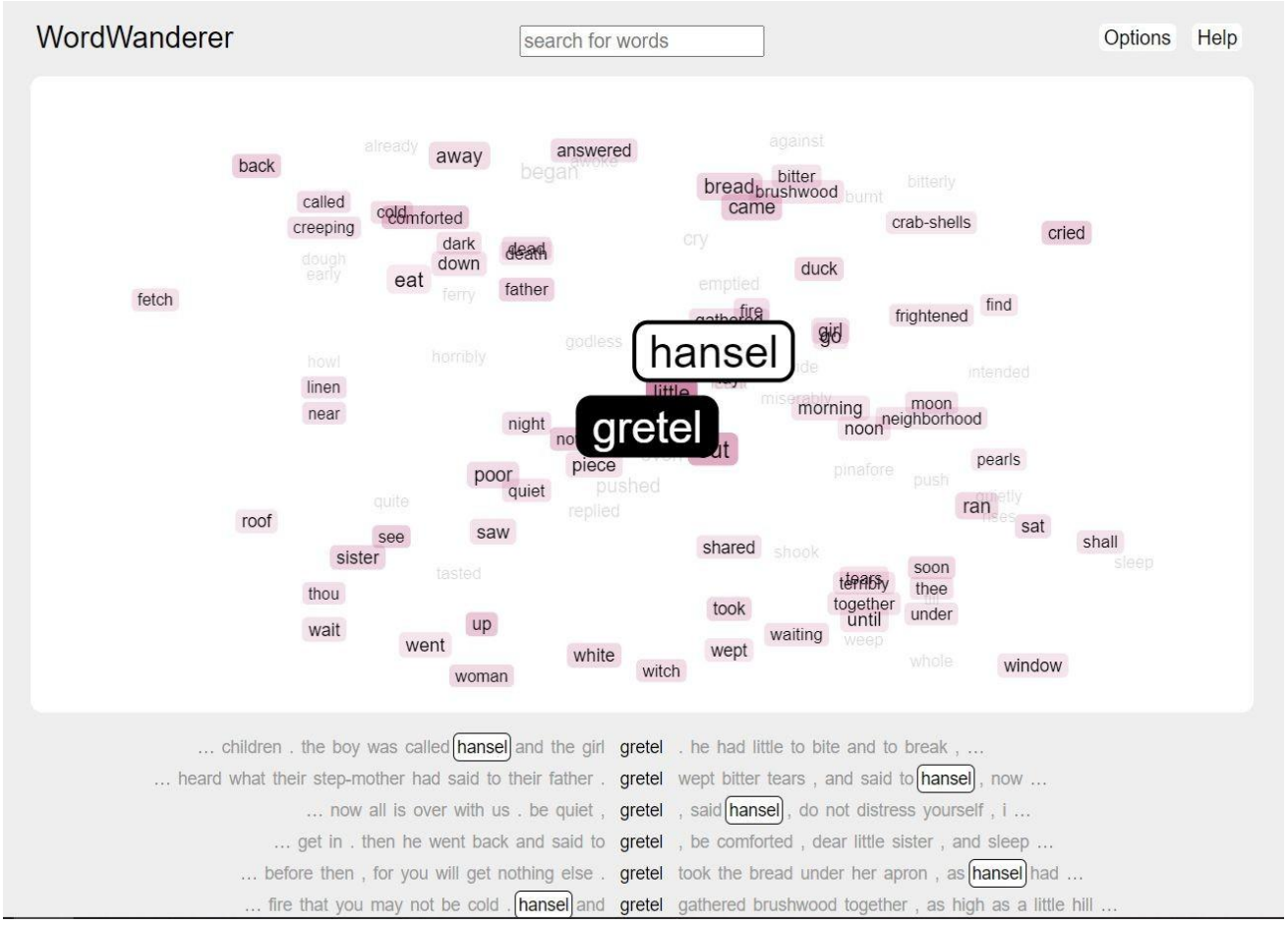


Рисунок 1.2 - Інтерфейс веб-сервісу WordWanderer

При натисканні на слово, воно виділяється для представлення контексту, що є візуалізацією конкордантності, що розміщує вибране слово (вузол) у центрі подання та розташовує колокати (пов’язані слова) горизонтально відповідно до їх сукупного відносного положення. Наприклад, слово “діти” частіше з’являється перед “лісом” у казці Гензель та Гретель і, отже, саме слово у тексті розташоване лівіше. Розмір шрифту коллоката представляє силу асоціації з вузлом, яка базується на кількості спільних появ (кількість разів,

коли слово зустрічається у тексті) та їх близькості в тексті: чим більше слово, тим сильніша асоціація між словами та вибраним терміном. Вертикальне позиціонування є алфавітним, а горизонтальне позиціонування у свою чергу представляє положення речень щодо виділеного слова.

Перетягування рядка між двома словами викликає подання одного з режимів, а саме режим порівняння. Цей режим базується на виділенні двох слів, відображаючи ті слова, які є суміжними з обома виділеними словами. Хоча розмір шрифту базується на середньому значенні двох значень асоціації, різниця в асоціації між двома виділеними словами виражається горизонтальним розташуванням колокатів. Слова, які більше пов'язані, наприклад, з лівим словом, розташовані більше вліво і навпаки, пов’язані з правим словом у правому боці. Наприклад, слово “жінка” з’являється лівіше, і це вказує на те, що вона сильніше пов’язана з “дітьми”, ніж з “лісом”. Горизонтальне положення представляє різницю в асоціації між двома виділеними словами.

WordWanderer з одного боку не має фіксованого обмеження як такого для заданої частоти або сили асоціації, а з іншого боку, краще сказати, існує максимальна кількість слів, які можуть відображатися на основі макета. Хмарний режим, наприклад, показує 300 найбільш частих слів, тоді як контекстний та порівняльний режими показують лише 100 слів з найбільш сильними зв’язками.

По суті, мінімальна частота або міра асоціації динамічна і залежить лише від вибраного тексту. Кольорова гама у цьому прикладі системи не є такою обширною, але візуально інтерфейс легкий для розуміння [4].

1.1.3 Система візуалізації та пошуку інформації в текстових документах WordTree

WordTree, або дерево слів (рисунок 1.3) реалізую нову техніка візуалізації та пошуку інформації в текстових документах. Дерево слів, як можна перекласти дану систему, є графічною версією традиційного методу keyword-incontext (KWIC) і забезпечує швидкі запити та дослідження певного текстового

контексту. Дерево слів розміщує слова, що слідують за певним пошуковим терміном, у структурі в формі дерева і використовує її для просторового розташування цих слів у вигляді дерева слів. Усі ці прості прийоми взаємодії дозволяють користувачу вивчити методи використання певного слова чи фрази в тексті, зважаючи на його закономірності та деталі у тому чи іншому тексті.

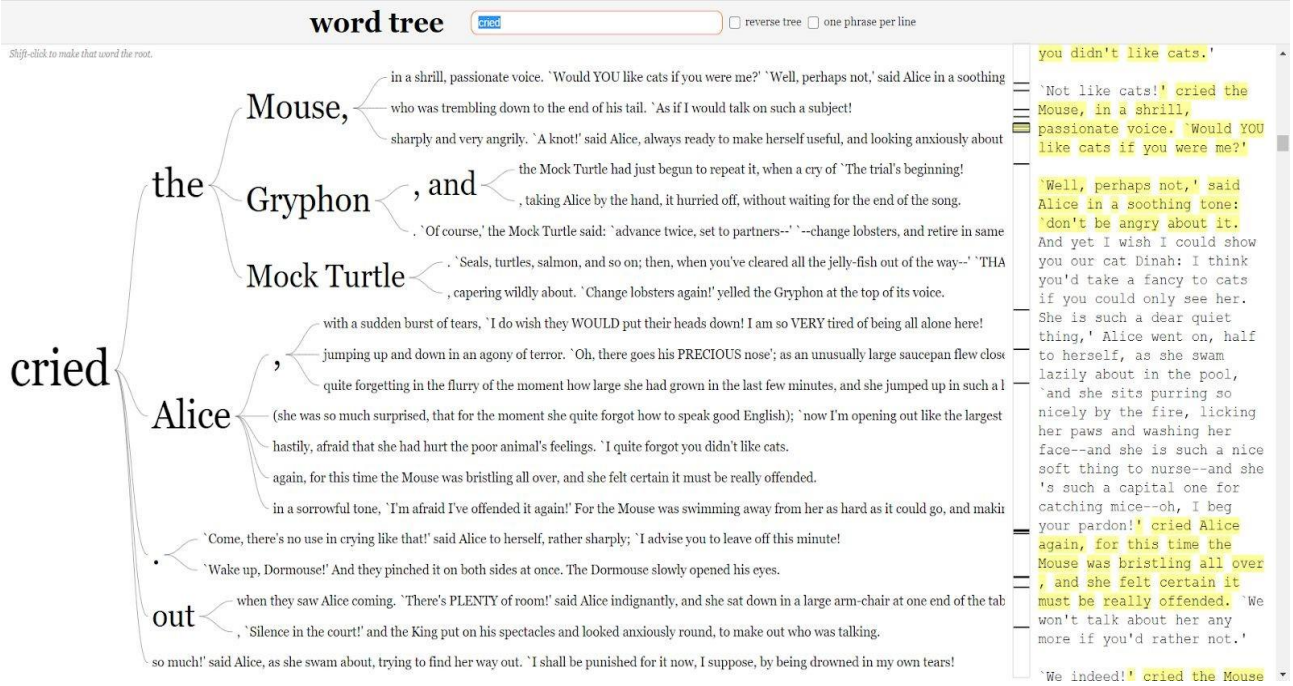


Рисунок 1.3 - Інтерфейс веб-сервісу WordTree

Дерево слів реалізує інтерактивну форму техніки KWIC. Сама ця система базується на даній техніці, проте вона містить покращення звичайного алгоритму.

По-перше, вона має візуальний дизайн, що дозволяє легко помітити повторення в контекстуальних словах, які у свою чергу слідують за фразою послідовно.

По-друге, дизайн інтерфейсу системи робить доволі простою та зручною структуру самого природного дерева контексту.

По-третє, це дає прості способи подальшого вивчення контексту.

Відображення KWIC можна вважати замаскованим деревом. На рисунку 1.4 показані слова, що стоять за пошуковим терміном «if love» (з англійської «якщо кохання») у п’єсі «Ромео та Джульєтта». Якщо центральним кореневим

вузлом брати пошуковий термін, то усі наступні слова визначають гілки. У цьому випадку, оскільки «if love» завжди супроводжується «be» (з англійської «бути»), воно має лише один дочірній вузол, що відповідає цьому слову. Однак у вузла «be» є двоє дітей, по одному на кожне з двох різних слів, що слідують за ним: «rough» (з англійської «жорстокий») та «blind» (з англійської «сліпий»). Продовжуючи таким чином, можна визначити деревну структуру, яка описує всі способи використання шуканого терміну.

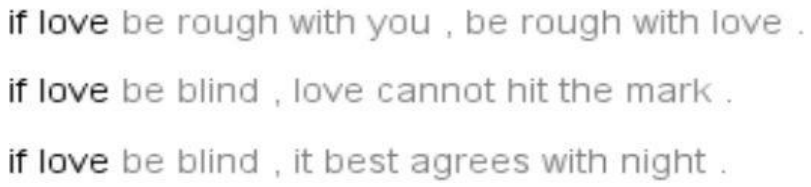


Рисунок 1.4 - Всі спогади «if love» в Ромео і Джульєтті

Ця структура не є новою. Основна ідея, яка називається суфіксальним деревом, впродовж десятиліть була складовою частиною алгоритмів обробки рядків. Однак, незважаючи на свою популярність серед розробників алгоритмів, дерева суфіксів не використовуються як загальний механізм візуалізації результатів пошуку [5].

1.1.4 Система семантичного аналізу російськомовних текстів Istio

Семантичний аналіз тексту від Istio оцінює ~~його~~ насиченість тексту ключовими словами, а також «водність» та «заспамленість». Пошукові системи визначають якість і релевантність текстового контенту за словами і словосполученнями, з яких він складається. Якщо в тексті досить тематичних ключових фраз, то пошукові системи оцінять його як хороший. Статті, в яких переважає «вода» і мало ключових слів, не потрапляють на перші сторінки видачі. Контент, перенасичений ключовими словами, відноситься до переспаму, його пошукові системи показують рідко.

Сервіс Istio показує:

- щільність ключових слів, їх процентне співвідношення в ядрі і в тексті;
- обсяг статті: кількість слів і символів (з пробілами і без);

- словник: загальна кількість одиниць, словник ядра;
- частотність слів, виводить топ-10 найбільш уживаних;
- мову статті і приблизну тематику;
- відсоток «води».

Для зручності користувачів сервіс підсвічує ключові слова (рисунок 1.5) і «воду», створює наочну карту частотних слів (рисунок 1.6). У верхній частині сторінки відображається кількість і процентне співвідношення ключових фраз в ядрі і тексті. У пункті «Карта тексту» користувач може наочно побачити, які одиниці часто повторюються в статті. Можна візуально оцінити, на якій відстані знаходяться повторювані фрази.

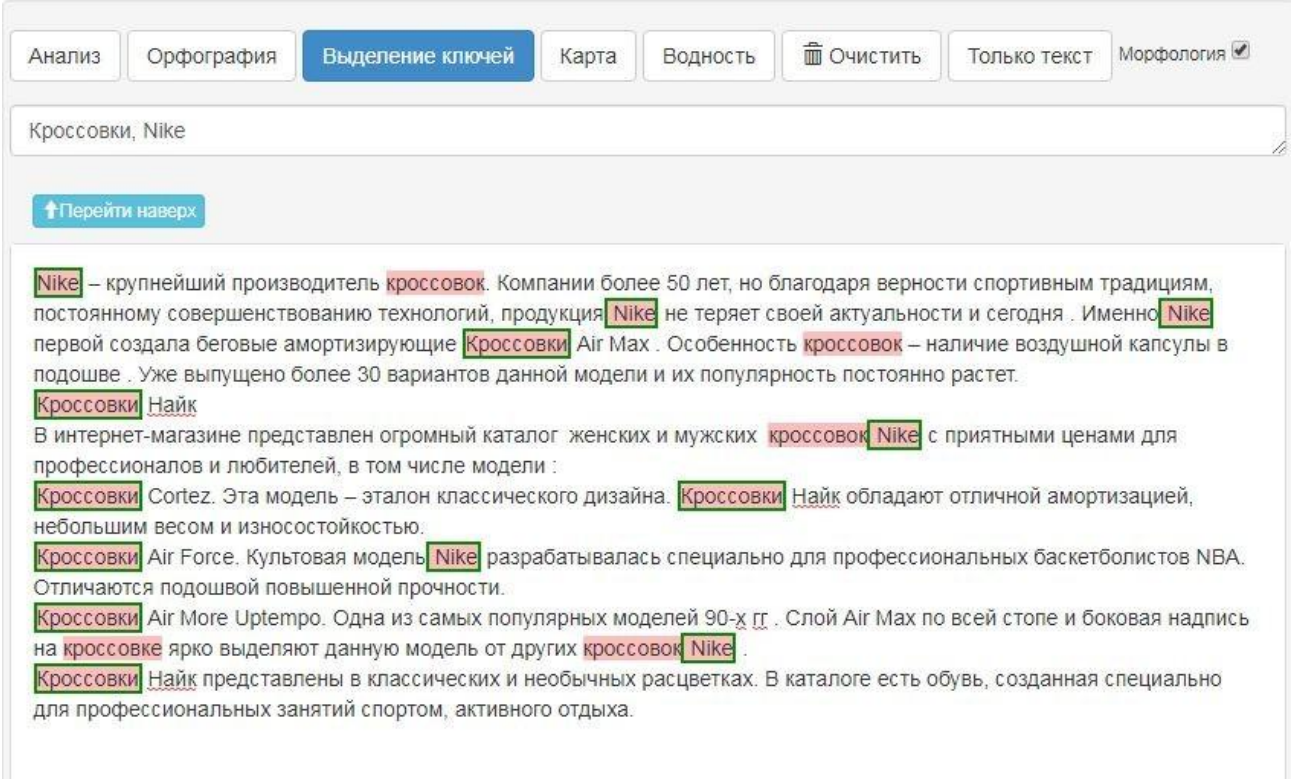


Рисунок 1.5 - Виділення ключів у тексті у веб-сервісі Istio

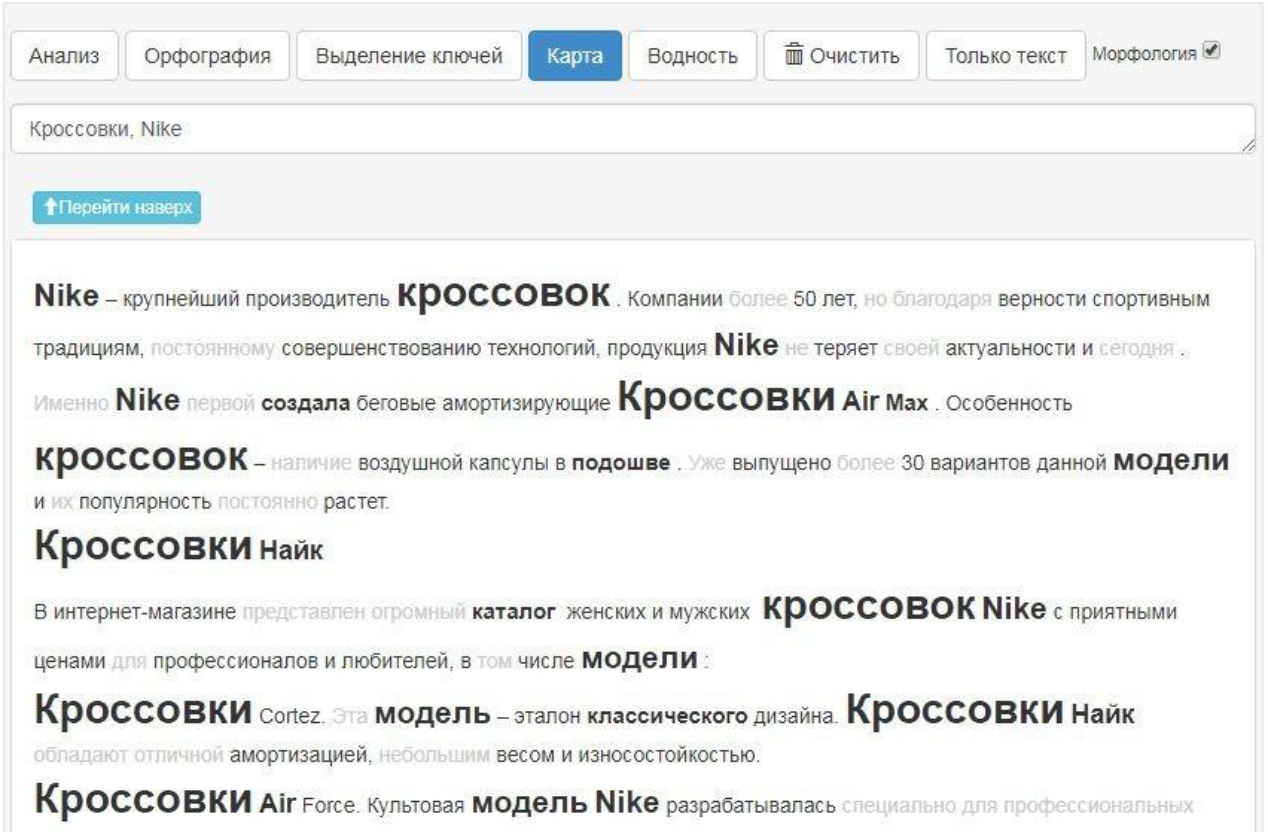


Рисунок 1.6 - Карта частотних слів у тексті у веб-сервісі Istio

Аналіз «води» тексту відображає наявність в статті стоп-слів, фразеологізмів, сполучних одиниць, які не несуть змістового навантаження. Якщо їх видалити, контент не втратить сенсу і стане якіснішим. Це одиниці з необ'єктивною оцінкою, що не несуть конкретної інформації, а також підсилювачі. Виділені фрази сервіс рекомендує видалити або замінити. Невелика кількість незначущих одиниць є природною для будь-якого тексту, проте показник потрібно зводити до мінімуму, щоб отримувати якісний вміст без «води» для пошукової системи.

1.2 Засоби семантичного аналізу україномовних текстів

Хоча програмних засобів підтримки текстів українською мовою не є багато на сьогоднішній день, проте деякі з них уже є доволі значними. Багато з них можна знайти на сайті lang-uk [6], що є відкритою для приєднання спільнотою фахівців в галузі комп'ютерної обробки текстів (програмістів,

лінгвістів, дослідників). Ця спільнота побудована на єдиних принципах і займається підтримкою наявних і розвитком нових проектів по збору українських корпусів та інших текстових даних. Основними напрямками їх роботи є збір і публікація корпусів та інших наборів текстових даних українською мовою, створення на основі цих даних моделей для вирішення прикладних завдань обробки українських текстів та імплементація цих моделей у наборі публічно-доступних мікросервісів. Серед їх активних проектів можна виділити збирання текстів для корпусу БрУК, збирання максимально можливого корпусу українських текстів з різноманітних джерел, побудова на основі цього корпусу моделей векторного представлення слів, створення анотованого корпусу для задачі побудови зв'язків між сутностями.

Також, цікавим проектом є морфосинтаксовий аналізатор української мови [7], що дозволяє здійснювати автоматизований морфологічний аналіз лексем (рисунок 1.7). Ця модель навчена на золотому стандарті і має точність морфологічних рис 91.6%, а синтетичних зв’язків – 81.7%. Розроблене також API, що на вхід отримує текст в UTF-8 і розміром щонайбільше 1 мегабайт, і на вихід дає повний його розбір (поділ на слова, речення, морфо і синтетичний аналіз).

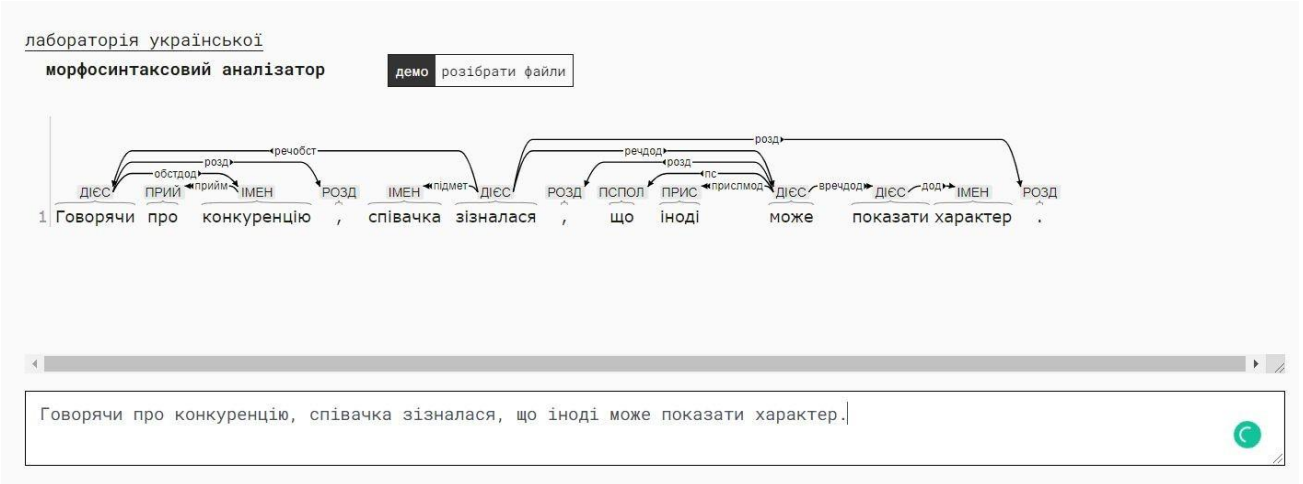


Рисунок 1.7 - Морфосинаксичний аналіз україномовного тексту

Інший варіант морфоаналізатора – pymorphy2, що написаний на мові Python і розміщений у відкритому репозиторії github. Він дозволяє приводити

слово до нормальної формі, наприклад "люди -> людина", або "гуляв -> гуляти". Також є можливість поставити слово в потрібну користувачу форму, наприклад ставити у множину, міняти відмінок слова і тощо. Pymorphy2 також дозволяє отримувати граматичну інформацію про слово – його число, рід, відмінок, частина мови тощо. При роботі pymorphy2 використовується словник OpenCorpora, а для незнайомих слів будуються гіпотези. Бібліотека є досить швидкою, зараз швидкість роботи складає від декількох тисяч слів за секунду до більше ніж 100 тисяч слів за секунду в залежності від виконуваної операції, інтерпретатора та встановлених пакетів.

І це лише декілька прикладів амбітних проектів з аналізу української мови які набирають оберту за останні роки. Оскільки технології з обробки природньої мови набувають популярності, в тому числі і в Україні, їх кількість буде лише зростати.

1.3 Методи тематичного моделювання природно-мовних текстів

Тематичне моделювання - це спосіб побудови моделі корпусу текстів, що відображає перехід від сукупності документів, сукупності слів в документах до набору тем, що характеризують зміст даних документів. Найбільш популярні зараз методи тематичного моделювання можна розділити на дві основні групи - алгебраїчні та ймовірнісні. До алгебраїчних моделей належать стандартна векторна модель тексту VSM і латентно-семантичний аналіз LSA, а серед імовірнісних найбільш популярними є імовірнісний латентно-семантичний аналіз PLSA і латентне розміщення Діріхле LDA [8].

1.3.1 Векторна модель текстів

Векторна модель текстів (англійською VSM – Vector Space Model) - це спосіб представлення колекції документів у вигляді векторів із загального для всієї колекції векторного простору. Дана модель використовується для вирішення безлічі завдань швидкого аналізу документів, а також для складання

таблиць пошуку, класифікації та кластеризації, і виступає як основа для безлічі інших алгоритмів. У даній моделі, документ розглядається як неврегульована безліч термів - слів і додаткових елементів, з яких складається текст. Термами можуть виступати як слова, так і їх комбінації, так звані н-грами, документами - в ідеалі: набори тематично однорідних текстів, або просто будь-який бажано об'ємний текст довільно розбитий на шматки, наприклад абзаци.

Насамперед необхідно провести попередню обробку всіх документів. Перш за все це видалення всіх знаків пунктуації та спеціальних знаків. Також, необхідно виключити так звані стоп-слова або шумові слова — це слова, які не несуть змістового навантаження, тому їх користь та роль для пошуку не суттєва. Наприклад, всі прийменники, суфікси, дієприкметники, вигуки, цифри необхідно виключити. Список шумових слів для української мови можна знайти в репозиторії українського аналізатору пошукової системи Lucene [9].

Далі необхідно провести операцію стемінгу. Цей крок можна опустити, якщо тексти є англомовними в силу того, що кількість варіацій тієї чи іншої словоформи в англійській мові значно менше ніж в українській. У випадку з українською мовою, пропускати цей крок не варто, тому що це може призвести до істотної деградації результатів. Одним з популярних алгоритмів є алгоритм Мартина Портера, який втілив його в проекті Snowball для низки індоєвропейских мов, в т.ч. для російської, але не для української. Варіанти стемінгу для української мови існують і використовуються у складі комерційних пошукових систем. На жаль, наразі відсутня вільна реалізація подібних алгоритмів. Слід зазначити, що певні кроки у цьому напрямку вже зроблені, наприклад стемінг за словником в українському аналізаторі для вже згаданої пошукової системи Lucene [9], або стеммер базований на алгоритмі Миколи Сеника [10].

Наступним кроком є подання терм-документної матриці, що описує частоту термінів, які зустрічаються в колекції документів. Рядки відповідають документам в колекції, а стовпці відповідають термінам. Значенням клітинки даної матриці є вага даного слова в документі, спосіб обчислення якого

змінюється в залежності від алгоритму. Першим кроком до цього є складання частотної матриці індексованих слів. У кожній клітинці матриці вказано скільки разів слово зустрічається у відповідному документі.

Слово, що зустрічається в більшості документів, може не відповідати справжнім значенням документа, тоді як рідкісні терми можуть визначати релевантність документа. Це можна досягти за допомогою методу «частота терму - зворотна частота документа» (TF-IDF), який надає більші значення термам, які зустрічаються частіше в документі, але не часто трапляються у всіх інших документах, і відповідно навпаки. Для обчислення IDF використовується формула (1.1) – логарифм зворотного дробу між кількістю документів і частотою даного документу [11].

— , (1.1)

де IFD_i – значення IDF для i-го терму;

m – загальна кількість унікальних термів в корпусі;

N – загальна кількість документів в корпусі;

df_i – кількість документів що містять i-ий терм.

Коли будуємо терм-документну матрицю, кожен вектор документа представляє точку в векторному просторі. Математично близькість між двома векторами можна розрахувати шляхом обчислення скалярного добутку між ними. Таким чином, щоб знайти відповідний документ для запиту, необхідно обчислити показник подібності між кожним вектором документа і вектором терму запиту, застосовуючи косинусну схожість.

Схожість між будь-якою комбінацією термів і документів найчастіше обчислюють саме за допомогою цього методу, однак на практиці кращий результат дає обчислення схожості за допомогою коефіцієнта кореляції Пірсона. Векторна модель текстів досить популярна для вирішення завдань порівняння текстів між собою, однак в початковому варіанті працює недостатньо швидко для великих обсягів документів, а також займає досить

багато пам'яті.

1.3.2 Латентно-семантичний аналіз

Розвитком попереднього методу є латентно-семантичний аналіз (англійською LSA - Latent Semantic Analysis). Латентно семантичний аналіз - це статистичний метод обробки текстової інформації на природній мові, що дозволяє визначати взаємозв'язок між колекціями документів і термами, що в них зустрічаються. В основі цього методу лежить принцип факторного аналізу, зокрема виявлення латентних зв'язків досліджуваних явищ і об'єктів. При класифікації і кластеризації документів, даний метод дозволяє отримати контекстно-залежні значення лексичних одиниць.

Основна ідея латентно-семантичного аналізу полягає в наступному: якщо у вихідному ймовірнісному просторі, що складається з векторів слів між двома будь-якими словами з двох різних векторів може не спостерігатися ніякої залежності, то після деякого алгебраїчного перетворення даного векторного простору ця залежність може з'явитися, причому величина цієї залежності визначатиме силу асоціативно-семантичного зв'язку між цими двома словами. Тобто цей аналіз дозволяє виявляти значення слів з урахуванням контексту їх використання шляхом обробки великого обсягу текстів [14].

Модель представлення тексту, що використовується в латентно-семантичному аналізі, багато в чому схожа із сприйняттям тексту людиною. Наприклад, за допомогою цього методу можна оцінити текст на відповідність заданій темі або провести реферування [15]. LSA відображає документи і окремі слова в так званий «семантичний простір», в якому і проводяться всі подальші порівняння. При цьому робляться такі припущення:

- документ це просто набір слів, а їх порядок у документах ігнорується, важливо лише скільки разів слово зустрічається в документі;
- семантичне значення документа визначається набором слів, які йдуть разом;
- кожне слово має єдине значення, що є спрощенням, але саме воно дозволяє вирішити багато проблем.

Основний алгоритм даного методу можна розділити на чотири етапи. Перші три етапи аналогічні до методу VSM: попередня обробка, знаходження ваг слів, наприклад, за допомогою алгоритму TF-IDF та побудова вагової матриці. Останнім етапом є розкладання матриці методом сингулярного розкладання (Singular value decomposition, SVD). Сингулярне розкладання - це математична операція, за допомогою якої матрицю розкладають на 3 складових. Сингулярне розкладання можна уявити у вигляді формули (1.2).

$$A = U \times S \times V^T, \qquad (1.2)$$

де – вихідна матриця;

і – ортогональні матриці;

– діагональна матриця, значення, на діагоналі якої називаються сингулярними коефіцієнтами матриці і які упорядковані за спаданням.

Сингулярне розкладання дозволяє виділити ключові складові вихідної матриці. Основна ідея LSA полягає в тому, що якщо в якості матриці використовувалася терм-документ матриця, то матриця *, що містить тільки перших лінійно незалежних компонент, відображає основну структуру різних залежностей, присутніх у вихідній матриці. Структура залежностей визначається ваговими функціями термів [13].

Основними перевагами даного методу можна вважати високу якість визначення тематик в разі, якщо корпус текстів досить великий, а також можливість знаходження неочевидних семантичних залежностей між словами. До недоліків даного алгоритму відносяться висока обчислювальна складність і низька швидкість роботи, що вимагає повторного обчислення всіх метрик для всього корпусу в разі додавання нового документа, а також високі вимоги до корпусу, який повинен складатися з безлічі різноманітних за тематиками текстів.

1.3.3 Імовірнісний летентно-семантичний аналіз

Імовірнісний латентно-семантичний аналіз (англійською pLSA - Probabilistic Latent Semantic Analysis) — це статистичний метод аналізу кореляцій двох типів даних. У загальному, даний метод є розвитком латентно-семантичного аналізу, однак на відміну від свого попередника, який за своєю суттю був алгоритмом побудови векторного уявлення з наступним зниженням його розмірності, імовірнісний латентно-семантичний аналіз заснований на змішаному розкладанні і використанні імовірнісної моделі, що дозволяє більш якісно та чіткіше визначати можливі тематики документів. PLSA у свою чергу, використовує імовірнісний метод замість SVD. Основна ідея полягає в знаходженні вірогідної моделі з прихованими темами, яка може генерувати дані, які спостерігаються в нашій терм-документній матриці. Зокрема, ~~нам~~ потрібна модель P(D, W), така, що для будь-якого документа d і слова w, P(d, w) відповідає цьому запису в терм-документній матриці [17]. Припустимо, що кожен документ складається з набору тем, а кожна тема - з набору слів, тоді PLSA додає імовірнісний фактор до припущень:

- тема z присутня в документі d з ймовірністю P(z | d);
- при заданій темі z слово w взято з z з ймовірністю P(w | z).

Формально, загальну ймовірність, щоб побачити даний документ і дане слово разом, можна обчислити за формулою:

, *(1.3)*

де *P(D)*, *P(Z|D)*, та *P(W|Z)* – параметри моделі;

P(D) – визначається безпосередньо зі значень корпусу;

P(Z|D) та *P(W|Z)* – значення моделі, які моделюються як мультиноміальні розподіли, і їх можна навчити, використовуючи алгоритм очікування-максимізації (EM).

Не вдаючись до повної математичної обробки алгоритму, EM є методом пошуку найбільш вірогідних оцінок параметрів для моделі, яка залежить від неспостережуваних, прихованих змінних (у нашому випадку, тем).

Інтуїтивно зрозуміло, що права частина цього рівняння показує наскільки ймовірно, що відобразиться певний документ, а потім, грунтуючись на розподілі тем цього документа, наскільки ймовірно знайти певне слово в цьому документі. Цікаво, що P(D, W) може бути еквівалентне параметризації з використанням іншого набору з 3 параметрів:

Аналізуючи цю формулу, можна зрозуміти цю еквівалентність, розглядаючи саму модель як певний генеративний процес. У першій параметризації все починалось з документа за допомогою P(d), а потім генерувалась тема за допомогою P(z | d), а потім генерувалось слово за допомогою P(w | z). У цій параметризації все починається з теми з P(z), а потім незалежно генерується документ з P(d | z) і словом з P(w | z).

Причина, через яку нова параметризація настільки цікава та незвичайна, полягає в тому, що можна бачити пряму паралель між моделлю PLSA і моделлю LSA: де ймовірність теми P(Z) відповідає діагональної матриці ймовірностей єдиної теми S, ймовірності нашого документа враховуючи, що тема P(D | Z) відповідає матриці тематики документа U, а ймовірність слова в даній темі P(W | Z) відповідає матриці термінів теми V [9]. До переваг даної моделі, щодо інших схожих за функціоналом алгебраїчних, можна віднести можливість знаходження ймовірності відносини кожного документа до кожної з представлених тем, з подальшим угрупуванням, що є досить трудомістким завданням для алгоритму LSA. Недоліками даної моделі є недоліки, властиві і LSA, до яких відноситься і необхідність перебудови всієї моделі в разі додавання нового документа, а також лінійна залежність кількості параметрів від кількості документів [16].

1.3.4 Латентне розміщення Діріхле

Латентне розміщення Діріхле (англійською LDA - Latent Dirichlet

Allocation) на мові оригіналу - вживана в інформаційному пошуку породжувальна модель, що дозволяє пояснити результати спостережень за допомогою деяких неявних (латентних) груп.

Дана модель є розширенням іншої, схожої за властивостями моделі PLSA, і усуває основні її недоліки шляхом використання розподілу Діріхле, в результаті чого набір тематик виходить більш конкретний і чіткий. Дана модель дозволяє уникнути дуже багатьох недоліків своєї попередньої версії PLSA, таких як:

- «перенавчання» - виникає тоді, коли модель є занадто складною, а саме такою, що має занадто багато параметрів відносно числа спостережень;
- відсутність закономірності при генерації документів з набору отриманих тем, що значно покращує підсумкову фінальну вибірку.

Розглянемо актуальний приклад порівняння імовірнісних розподілів тематичних сумішей. Припустимо, що корпус, який розглядається, має документи з трьох дуже різних предметних областей. Якщо потрібно змоделювати це, то тип дистрибутива, який потрібен, буде дуже сильно зважувати одну конкретну тему і не надасть значення іншим. Якщо є 3 теми, то будуть такі розподіли ймовірностей:

- суміш X: 90% тема A, 5% тема B, 5% тема C;
- суміш Y: 5% тема A, 90% тема B, 5% тема C;
- суміш Z: 5% тема A, 5% тема B, 90% тема C.

Якщо ж намалювати випадковий розподіл ймовірностей з цього розподілу Діріхле, параметризовані великими вагами по одній темі, ймовірно, отримається розподіл, який сильно нагадує або суміш X, суміш Y або суміш Z. Було б дуже малоймовірно, щоб розподіл, який становить 33% це тема A, 33% це тема B і 33% це тема C. По суті, це те, що забезпечує розподіл Діріхле: спосіб вибірки розподілів ймовірностей певного типу.

У модель PLSA відбирається документ, потім тема, заснована на цьому документі, потім слово, засноване на цій темі. З розподілу Діріхле Dir(α) береться випадкова вибірка, що представляє розподіл чи суміш тем

конкретного документу (позначимо його θ). З θ вибирається конкретна тема Z на основі розподілу. Далі, з іншого розподілу Діріхле Dir(β) обирається випадкова вибірка, що представляє розподіл слів по темі Z (позначимо його φ). З ф вибирається слово w. LDA зазвичай працює краще, ніж PLSA, тому що він може легко узагальнювати нові документи. У PLSA ймовірність документа є фіксованою точкою в наборі даних, відповідно якщо не бачити документ, то немає цієї точки даних. У LDA набір даних служить навчальними даними для розподілу Діріхле за темами документів. Якщо не бачити документ, можна легко взяти зразок з дистрибутива Діріхле і рухатися далі [12].

В результаті розгляду деяких основних методів тематичного моделювання можна прийти до висновку, що методи, засновані на імовірнісних моделях по своїй суті найкращим чином придатні для вирішення поставленого завдання, однак, вимагають високих обчислювальних витрат при реалізації у своєму початковому вигляді. Метод LDA є найбільш складним, і при цьому дозволяє досягти найкращих результатів, і уникнути основних недоліків звичайного PLSA. Розглянуті методи легко застосовуються для аналізу україномовних текстів, адже на даний момент у світі існує достатній об’єм необхідних даних, реалізації алгоритмів, а також навчальних корпусів[18].

Висновки до розділу

У даному розділі були розглянуті різні популярні засоби семантичного аналізу текстів іноземними мовами, а також україномовних текстів. Засобів для англійської та російської мов значно більше, проте є корисні ресурси, які можна використовувати як базові при розробці моделі семантичного аналізу для української мови. Серед них такі сервіси як InfraNodus, WordWanderer, WordTree для англійської мови, та Istio для російської.

Серед існуючих засобів семантичного аналізу україномовних текстів було розглянуто морфоаналізатор pymorphy2, а також морфосинтаксичний аналізатор. Було вирішено використовувати відкриту бібліотеку pymorphy2 для удосконалення роботи розробленої системи.

Також було здійснено аналіз та порівняння методів тематичного моделювання корпусу текстів - а саме Vector Space Model, Latent Semantic Analysis, Probabilistic Latent Semantic Analysis та Latent Dirichlet Allocation.

2 УДОСКОНАЛЕНИЙ МЕТОД СЕМАНТИЧНОГО АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

2.1 Розроблення удосконаленого методу семантичного аналізу україномовних текстів

Розглянувши існуючі засоби та методи визначено, що результатом магістерської дисертації повинна стати модель, що проводить тематичне моделювання корпусу україномовних текстів. Модель повинна базуватись на вищезгаданих методах семантичного аналізу адаптованих під українську мову. Для збільшення функціональності модель повинна використовувати доступні існуючі засоби семантичного аналізу для української мови. Для підвищення ефективності необхідно оптимізувати збереження даних, які будуть повторно використовуватися в обчисленнях. Для покращення сприйняття результатів моделювання необхідно розробити візуальний графічний інтерфейс з інтуїтивним та простим відображенням результатів роботи моделі у вигляді графу термів. Діаграму потоків даних у системі можна побачити на рисунку 2.1.

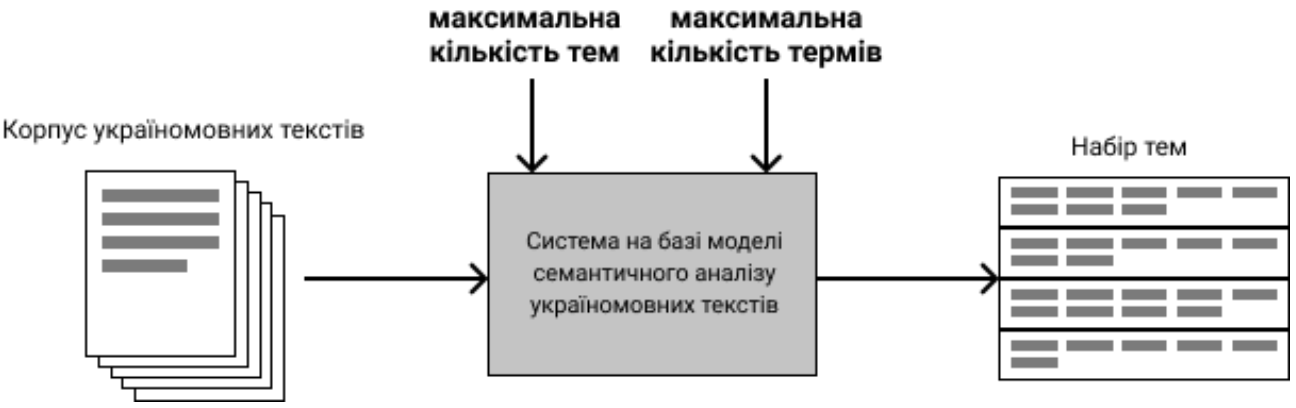


Рисунок 2.1 - Діаграма потоків даних у системі

На вхід моделі подається корпус україномовних текстів, кожен з яких повинен мати заголовок та контент, який буде проаналізований. Корпус повинен містити достатню кількість документів на різну тематику, щоб модель

могла виділити декілька різних тем. Корпус не повинен містити дублікати документів.

Також необхідно передати два параметри обмежувачі.

Перший параметр - це максимальна кількість тем, які необхідно отримати на виході. Оскільки модель не зможе визначити чи достатньо розподілити корпус на точну кількість тем, то користувач повинен встановити верхнє обмеження. Це не означає, що модель визначить рівно стільки тем, як і число в обмежувачі, оскільки деякі теми можуть включати в себе інші.

Другий - це максимальна кількість слів, які повинні містити всі теми в загальному. Цей параметр необхідний для візуалізації результату у вигляді графу, оскільки занадто складні структури можуть бути важкими для сприйняття.

На виході з моделі отримуємо певну кількість тем, які були виділені з корпусу текстів. Кожна тема - це множина термів, які відсортовані за релевантністю до цієї теми за спаданням.

Для внутрішніх обчислень був обраний метод латентно-семантичного аналізу. Основними перевагами даного методу можна вважати високу якість визначення тематик у разі, якщо корпус текстів досить великий, а також можливість знаходження неочевидних семантичних залежностей між словами. До недоліків даного алгоритму відносяться висока обчислювальна складність і низька швидкість роботи, що вимагає повторного обчислення всіх метрик для всього корпусу в разі додавання нового документа. Проте завдяки оптимізації збереження обчислень можна значно збільшити швидкість роботи системи, тому цей недолік не повинен стати на заваді роботи системи. Також, метод має високі вимоги до корпусу текстів, який повинен складатися з безлічі різноманітних за тематиками документів.

2.2 Приклад застосування розробленого методу

Розглянемо на детальних прикладах як працює модель. Вхідними даними взято корпус україномовних текстів, що складається з чотирьох коротких

документів, які містять спільні слова, щоб наочно показати результати (рисунок 2.2).

Рисунок 2.2 - Приклад корпусу документів на вхід до моделі

Насамперед необхідно провести попередню обробку всіх документів. Перш за все це видалення всіх знаків пунктуації та спеціальних знаків. Також необхідно виключити, так звані, стоп-слова або шумові слова — це слова, які не несуть змістового навантаження, тому їх користь та роль для пошуку не суттєва, наприклад всі прийменники, суфікси, дієприкметники, вигуки, цифри тощо. На рисунку 2.3 зображені документи корпусу після проходження попередньої обробки.

Рисунок 2.3 - Приклад корпусу документів після попередньої обробки

Після цього необхідно привести всі слова до нормальної форми, а також визначити частину мови цього терму за допомогою морфоаналізатора pymorphy2. Після нормалізації з масиву слів видаляються усі дублікати для отримання вектору термів для кожного документу (рисунок 2.4).

Рисунок 2.4 - Приклад векторів термів для кожного документу

На наступному кроці з векторів термів усіх документів формується bag of words – вектор, що містить унікальні терми, які зустрічаються в усіх документах. Після цього формується частотна матриця термів (Term Frequency) та вектор зворотної частоти документів (Inverse Document Frequency). Приклади цих обчислень можна побачити на рисунках 2.5 та 2.6.

	Д1	Д2	Д3	Д4
Гуртожиток	0	0	0	1/6
Дезинфікація	0	0	1/6	0
День	0	1/6	0	0
Київ	1/5	1/6	0	1/6
Коронавірус	0	0	1/6	1/6
Онлайн	0	1/6	0	0
Пожежа	1/5	0	0	0
Провести	0	1/6	1/6	0
Режим	0	1/6	0	0
Спалахнути	1/5	0	0	1/6
Торговий	0	0	1/6	0
Центр	1/5	0	1/6	0

Рисунок 2.5 - TF матриця для обраних вхідних даних

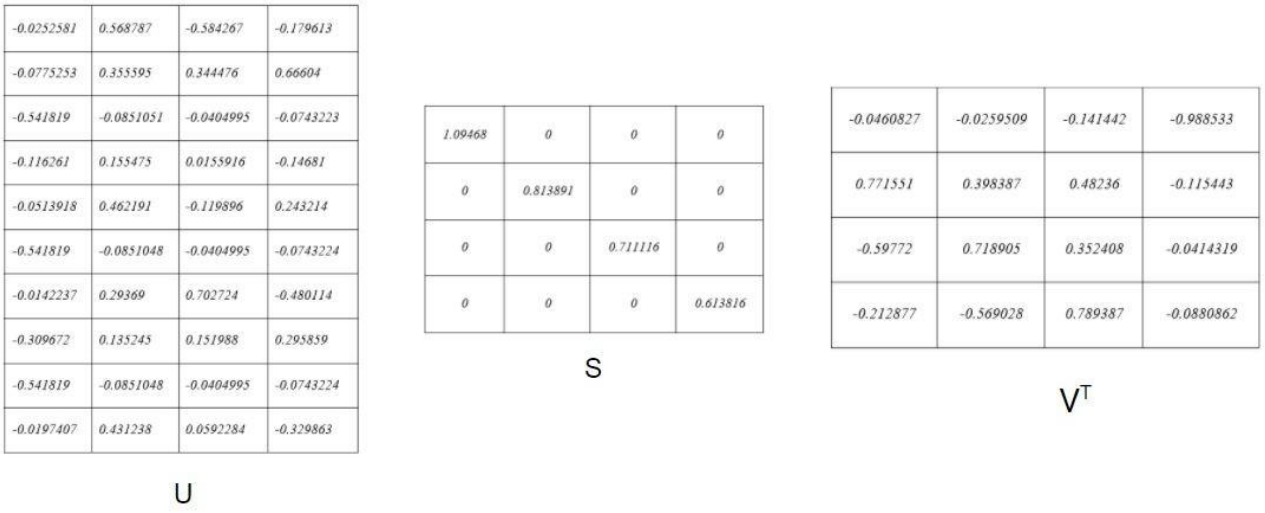
Рисунок 2.6 - IDF вектор для обраних вхідних даних

Добуток цього вектора та матриці дає TF-IDF матрицю (рисунок 2.7), яка надає більші значення термам, які зустрічаються частіше в документі, але нечасто трапляються у всіх інших документах, і відповідно навпаки.

	Д1	Д2	Д3	Д4
Гуртожиток	0	0	0	0.15
Дезинфікація	0	0	0.12	0
День	0	0.12	0	0
Київ	0.03	0.025	0	0.03
Коронавірус	0	0	0.06	0.08
Онлайн	0	0.12	0	0
Пожежа	0.15	0	0	0
Провести	0	0.06	0.06	0
Режим	0	0.12	0	0
Спалахнути	0.08	0	0	0.08
Торговий	0	0	0.12	0
Центр	0.08	0	0.06	0

Рисунок 2.7 - TF-IDF матриця для обраних вхідних даних

Наступним етапом є розкладання матриці методом сингулярного розкладання (Singular value decomposition, SVD). Сингулярне розкладання дозволяє виділити ключові складові вихідної матриці (рисунок 2.8).



-0.0252581	0.568787	-0.584267	-0.179613
-0.0775253	0.355595	0.344476	0.66604
-0.541819	-0.0851051	-0.0404995	-0.0743223
-0.116261	0.155475	0.0155916	-0.14681
-0.0513918	0.462191	-0.119896	0.243214
-0.541819	-0.0851048	-0.0404995	-0.0743224
-0.0142237	0.29369	0.702724	-0.480114
-0.309672	0.135245	0.151988	0.295859
-0.541819	-0.0851048	-0.0404995	-0.0743224
-0.0197407	0.431238	0.0592284	-0.329863

U

1.09468	0	0	0
0	0.813891	0	0
0	0	0.711116	0
0	0	0	0.613816

S

-0.0460827	-0.0259509	-0.141442	-0.988533
0.771551	0.398387	0.48236	-0.115443
-0.59772	0.718905	0.352408	-0.0414319
-0.212877	-0.569028	0.789387	-0.0880862

V^T

Рисунок 2.8 - Компоненти сингулярного розкладу для обраних вхідних даних

Основна ідея полягає в тому, що якщо в якості матриці використовувалася TF-IDF матриця, то матриця *, що містить тільки перших лінійно незалежних компонент, відображає основну структуру різних залежностей, присутніх у вихідній матриці. Структура залежностей визначається ваговими функціями термів. Це означає, що передавши обмеження кількості тем як параметр отримується матриця тем *, в якій стовпці відповідають темам, а рядки термам. Значення матриці - це релевантність належності певного терму до певної теми. Нехай для цього прикладу *k* має значення 2, тоді для обчислення матриці * потрібно взяти перші 2 стовпці матриці *U*, підматрицю *S* розміром 2x2, і перші 2 рядки матриці (рисунок 2.9).

-0.0252581	0.568787	-0.584267	-0.179613
-0.0775253	0.355595	0.344476	0.66604
-0.541819	-0.0851051	-0.0404995	-0.0743223
-0.116261	0.155475	0.0155916	-0.14681
-0.0513918	0.462191	-0.119896	0.243214
-0.541819	-0.0851048	-0.0404995	-0.0743224
-0.0142237	0.29369	0.702724	-0.480114
-0.309672	0.135245	0.151988	0.295859
-0.541819	-0.0851048	-0.0404995	-0.0743224
-0.0197407	0.431238	0.0592284	-0.329863

U

1.09468	0	0	0
0	0.813891	0	0
0	0	0.711116	0
0	0	0	0.613816

S

-0.0460827	-0.0259509	-0.141442	-0.988533
0.771551	0.398387	0.48236	-0.115443
-0.59772	0.718905	0.352408	-0.0414319
-0.212877	-0.569028	0.789387	-0.0880862

V^T

Рисунок 2.9 - Компоненти сингулярного розкладу для обраних вхідних даних

Після перемноження цих матриць отримано 2 стовпці, значення яких будуть показувати приналежність певного терму до даної теми. На рисунку 2.10 можна побачити, як буде виглядати тематична матриця для даного прикладу.

	Тема 1	Тема 2
Гуртожиток	0.046	> 0.001
Дезинфікація	0.041	0.014
День	> 0.001	0.117
Київ	0.017	0.021
Коронавірус	0.043	0.002
Онлайн	> 0.001	0.117
Пожежа	0.046	> 0.001
Провести	0.016	0.066
Режим	> 0.001	0.117
Спалахнути	0.046	> 0.001
Торговий	0.041	0.014
Центр	0.043	0.002

Рисунок 2.10 - Тематична матриця *A з для обраних вхідних даних**

Після цього можемо співставити кожному значенню матриці слово, зробити сортування по релевантності та повернути набір тем, що складається з масиву термів (рисунок 2.11). Відмітимо те, що чим більший корпус і чим більше в ньому слів, тим більш “логічним” для людського сприйняття буде розподілення тем. Модель дає можливість отримати результати, що залежать від вхідних даних та параметрів налаштування, тому добитись найкращого для сприйняття результату для будь-якого корпусу можна лише проводячи емпіричні тести.

Рисунок 2.11 - Результат розподілення термів на теми для обраних вхідних даних

Також, кожен з термів пропускається через NER-аналізатор. Named-entity recognition (NER, з англійської - розпізнавання іменованих сутностей) це підзадача видобування інформації, яка дозволяє знайти і класифікувати іменовані сутності в заздалегідь визначені категорії, такі як імена людей, організації, місця, медичні коди, час, кількості, грошові значення, відсотки тощо. Так, завдяки цьому аналізатору можна отримати більше інформації про кожен терм (що є іменником) на виході. Результат роботи NER-аналізатора для прикладу можна побачити на рисунку 2.12.

Рисунок 2.11 - Результат роботи NER-аналізатора для обраних вхідних даних

Отримані дані можна візуалізувати у вигляді певної структури, наприклад графу, який наглядно демонструє важливість термів в документах і темах та їх зв’язки між собою. В даній роботі був реалізований графічний інтерфейс у вигляді веб-застосунку, де користувач може інтерактивно взаємодіяти з графом термів, списком документів та тем, а також змінювати вхідні значення моделі.

Висновки до розділу

У даному розділі був описаний функціонал моделі семантичного аналізу україномовних текстів. Для роботи моделі був розроблений модифікований метод аналізу, що базується на методі латентно-семантичного аналізу, а також поєднаний з морфоаналізатором pymorphy і NER-аналізатором. Було продемонстровано роботу моделі на реальному прикладі з відображенням усіх проміжних кроків та результатів.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для того щоб звичайний користувач міг взаємодіяти з системою семантичного аналізу що реалізує запропонований алгоритм і модель розроблений веб-застосунок з інтерактивним графічним інтерфейсом. Застосунок надає можливість користувачу проаналізувати обраний ним корпус україномовних текстів та відобразити результат у вигляді графу термів.

3.1 Використані технології

3.1.1 Технологія Node.js

Для реалізації програмного засобу була використана технологія Node.js і відповідно мова програмування JavaScript.

Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

- асинхронна одно-нитева модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- рушій JavaScript Google V8.

Платформа Node.js призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Платформа крім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських і серверних програм.

JavaScript — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування - імперативну та частково функціональну і деякі відповідні архітектурні властивості, зокрема динамічну та слабку типізацію, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу [19].

3.1.2 Мова Python

Python — це високорівнева, загальнодоступна і дуже популярна мова програмування загального призначення високого рівня. Ця мова (остання версія Python 3) використовується при веб-розробці, в розробці застосунків машинного навчання, а також у багатьох передових технологіях в галузі програмного забезпечення. Python дуже добре підходить для початківців, а також для досвідчених програмістів, що володіють іншими мовами програмування, такими як C++ та Java.

Python був створений Гвідо ван Россумом у 1991 році і в подальшому розробляється організацією Python Software Foundation. Ця мова була розроблена з акцентом на читабельність коду, а його синтаксис дозволяє програмістам висловлювати свої концепції меншою кількістю рядків коду. Завдяки цьому python дозволяє швидко працювати та ефективніше інтегрувати системи.

Python має менше кроків у порівнянні з Java та C і використовується у багатьох організаціях, оскільки підтримує декілька парадигм програмування, а також виконує автоматичне управління пам'яттю.

Основні переваги даної мови:

- наявність сторонніх модулів;
- велика бібліотека підтримки (NumPy для чисельних розрахунків, Pandas для аналізу даних тощо);

- має відкритий код та розвивається комуною програмістів;
- легкий до вивчення;
- зручні структури даних;
- мова високого рівня;
- динамічно типізована мова (не потрібно призначати тип даних змінним, на основі присвоєного значення python сам визначає його);
- об’єктно-орієнтована мова;
- портативний та інтерактивний;
- працює на різних операційних системах.

В даній системі мова python використовується для того, щоб взаємодіяти з морфологічним аналізатором pymorphy2, що працює з українською мовою і дозволяє приводити слово до нормальної формі (наприклад, "люди - людина", або "гуляв - гуляти"). Також він може поставити слово в потрібну форму (наприклад, у множину, міняти відмінок слова) і повертати граматичну інформацію про слово (число, рід, відмінок, частина мови і т.д.). При роботі використовується словник OpenCorpora, а для незнайомих слів будуються гіпотези. Бібліотека досить швидка, зараз швидкість роботи складає від декількох тисяч до більше ніж 100 тисяч слів за секунду (в залежності від виконуваної операції, інтерпретатора і встановлених пакетів). Споживання пам'яті - від 10 до 20 мегабайт [20].

3.1.3 Формат даних JSON

JSON (JavaScript Object Notation) — це формат обміну даним, який легко читати та писати людям, та легко аналізувати і генерувати програмам. Він заснований на підмножині стандарту мови програмування JavaScript ECMA-262 3-го видання у грудні 1999 року. JSON — це текстовий формат, який зовсім не залежить від мови, але використовує конвенції, знайомі програмістам сімейства мов C, включаючи C, C++, C #, Java, JavaScript, Perl, Python та багато інших. Ці властивості роблять JSON ідеальною мовою обміну даними.

JSON побудований на двох структурах. Перша — це колекція пар ім’я/значення. У різних мовах це реалізується як об’єкт, запис, структура, словник, хеш-таблиця, список ключів або асоціативний масив.

Друга — це впорядкований список значень. У більшості мов це реалізується як масив, вектор, список або послідовність.

Це універсальні структури даних, адже практично всі сучасні мови програмування підтримують їх в тій чи іншій формі. Зрозуміло, що формат даних, який може використовуватись в різних мовах програмування, також має базуватися на цих структурах. У JSON ці структури набувають форми об’єкту — невпорядкованого набору пар ім’я/значення. Об'єкт починається з лівої хвилястої дужки і закінчується правою. Кожне ім'я супроводжується двокрапкою а самі пари ім'я/значення розділяються комою.

Оскільки в системі не потрібно зберігати складних структур даних, а також потрібна взаємодія з декількома різними мовами програмування, то JSON ідеально підходить для передачі та зберігання даних [21].

3.2 Керівництво користувача

Система являє собою веб-застосунок – сайт, на якому користувач може проводити семантичний аналіз корпусів україномовних текстів. При завантаженні сторінки користувач бачить інтерфейс системи, яка відображає результат аналізу першого прикладу корпусу текстів. На рисунку 3.1 можна побачити, як система відображає результат для вхідних даних, що обрані за замовчуванням.

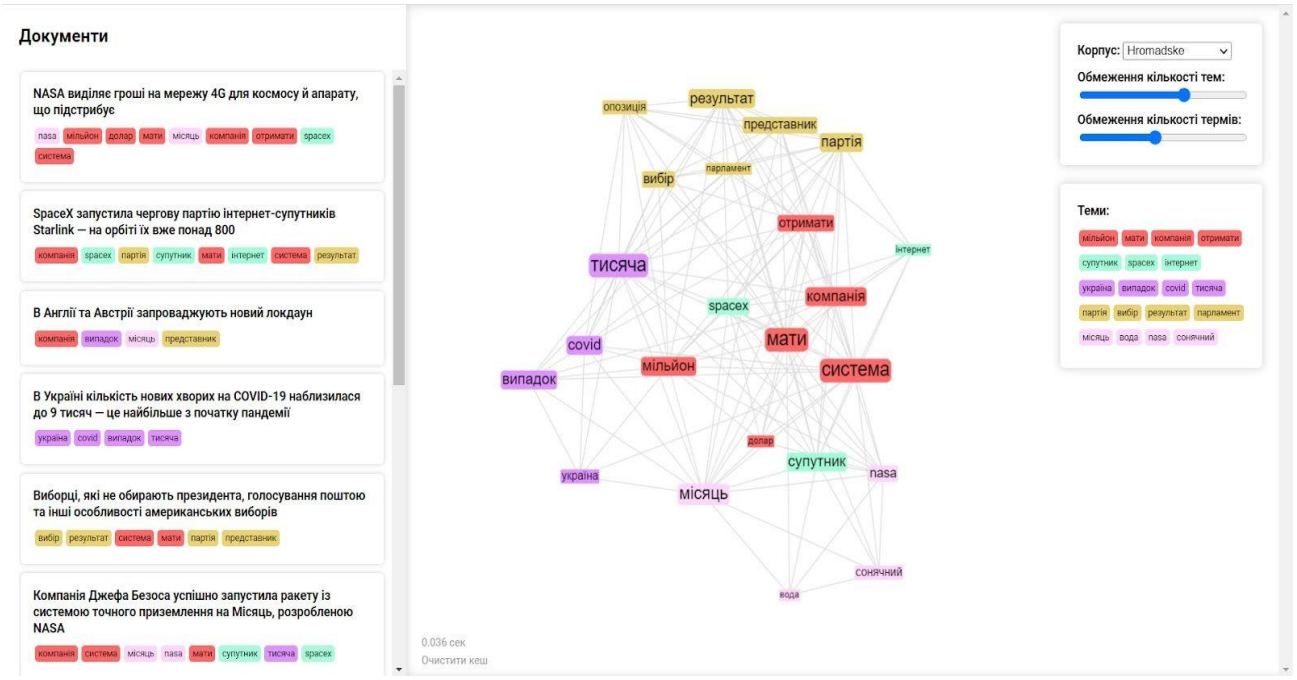


Рисунок 3.1 - Графічний інтерфейс розробленого веб-застосунку

У верхньому правому кутку є головна панель інтерфейсу, за допомогою якої користувач може налаштувати вхідні дані для моделі (рисунок 3.2). На ній користувач може обрати корпус текстів, які вже завантажені в систему, для проведення аналізу за допомогою випадаючого списку (рисунок 3.3). Також, ця панель містить два слайдери, за допомогою яких користувач може встановити обмеження на кількість тем, та на кількість термів, що будуть відображатись в результаті аналізу в інтерфейсі системи. За замовчуванням обраний перший корпус, що завантажений в систему, слайдер обмеження по кількості термів виставлений на максимальне значення - 50 термів, а обмеження по кількості тем - 40% від кількості документів в даному корпусі.

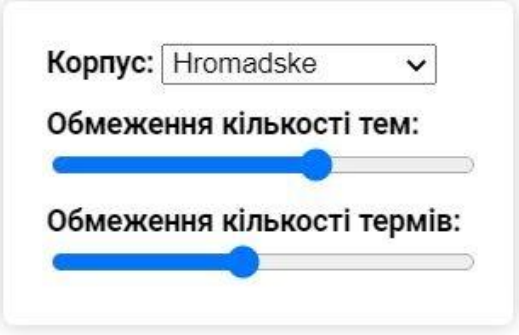


Рисунок 3.2 - Головна панель налаштувань інтерфейсу

Рисунок 3.3 - Випадаючий список для вибору корпусу текстів для аналізу

При будь-якій зміні налаштувань на головній панелі система змінює відображення результату аналізу в залежності від нових вхідних даних.

Одразу під головною панеллю налаштувань розміщене табло, що виводить список тем, які були отримані в результаті роботи системи (рисунок 3.4). Кожна тема має свій відповідний колір, для того щоб користувачу було зручно відрізняти терми, які належать даній темі, від інших. На таблі відображаються лише перші чотири найважливіших терми що описують дану тему.

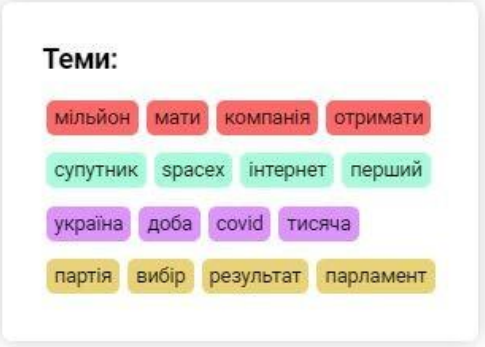


Рисунок 3.4 - Табло зі списком тем, отриманих в результаті аналізу

У лівій частині екрану є бічна панель, що виводить список усіх документів, що містить даний корпус текстів (рисунок 3.5). Документи виводяться відсортованими в алфавітному порядку в залежності від їх назви. Кожен документ - це картка, що має заголовок - назва документу, а також список термів, що є в тексті даного документу. Кожен терм має свій колір в залежності від того, до якої теми він належить. Також, терми сортовані в

порядку їх важливості в контексті даного документу.

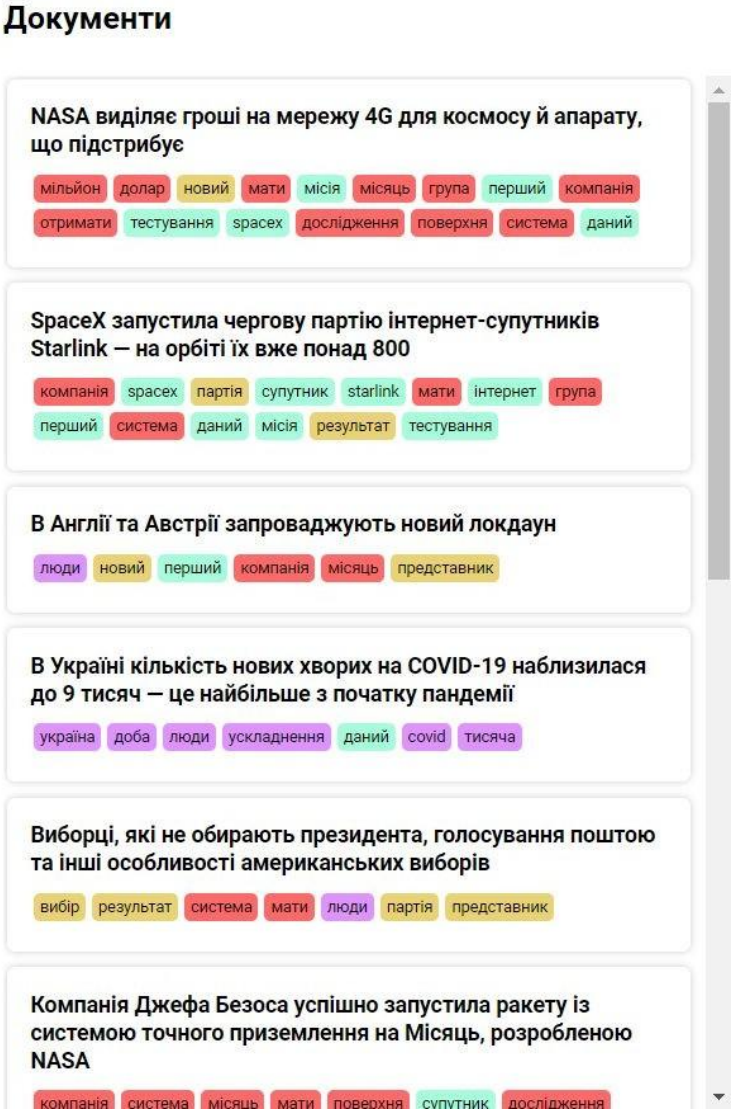


Рисунок 3.5 - Бічна панель зі списком документів

При натисканні на заголовок користувач може розгорнути карточку, та прочитати весь текст, що містить даний документ (рисунок 3.6). Щоб згорнути карточку, потрібно знову клікнути на заголовок документу.

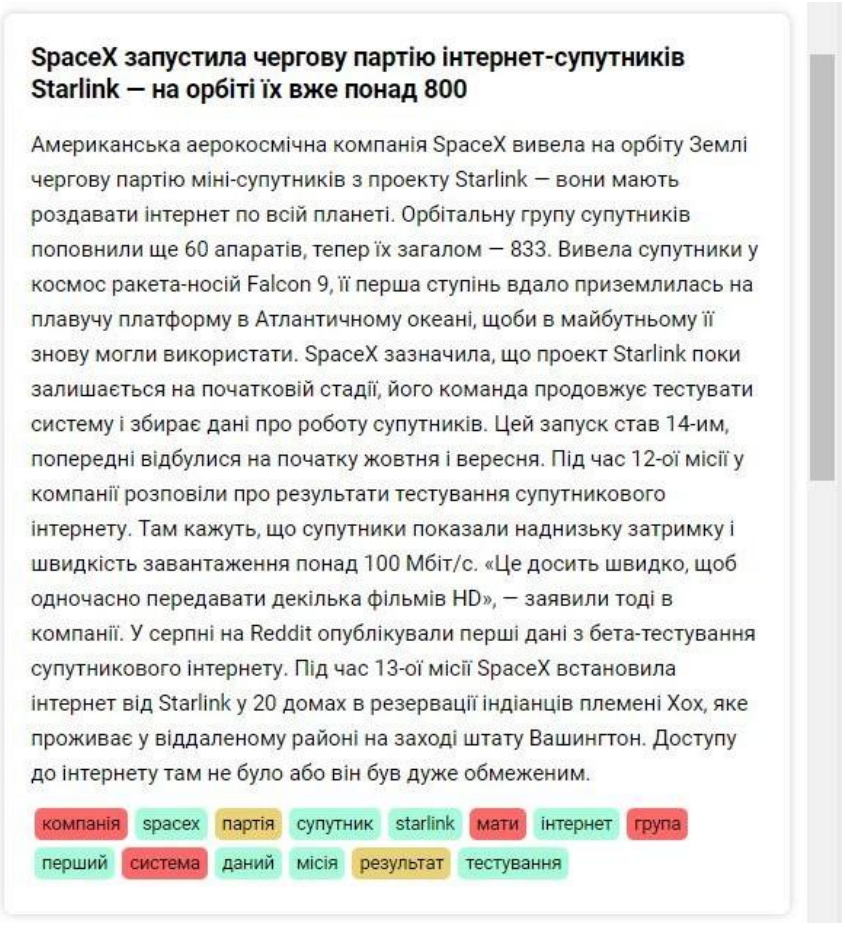


Рисунок 3.6 - Розгорнута карточка документу

Найважливіша частина графічного інтерфейсу - це граф термів, отриманий в результаті семантичного аналізу корпусу україномовних текстів (рисунок 3.7). Вузли в цьому графі являють собою терми, а ребра - це показник присутності двох термів в одному документі. У графічному відображенні кожен вузол має текст - значення терму що він репрезентує, а також свій колір, що залежить від теми, до якої належить даний терм. Розмір вузла залежить від того, в скількох документах зустрічається даний терм, тобто чим більше зв’язків, тим більший розмір.

Такий граф візуально демонструє терми в проаналізованому корпусі текстів так, що зрозуміла їх важливість, їх приналежність до певної теми, а також їх зв’язки з іншими термами. Сам граф є інтерактивний, тому користувач може легко змінити положення вузлів так, як йому буде легше сприймати інформацію.

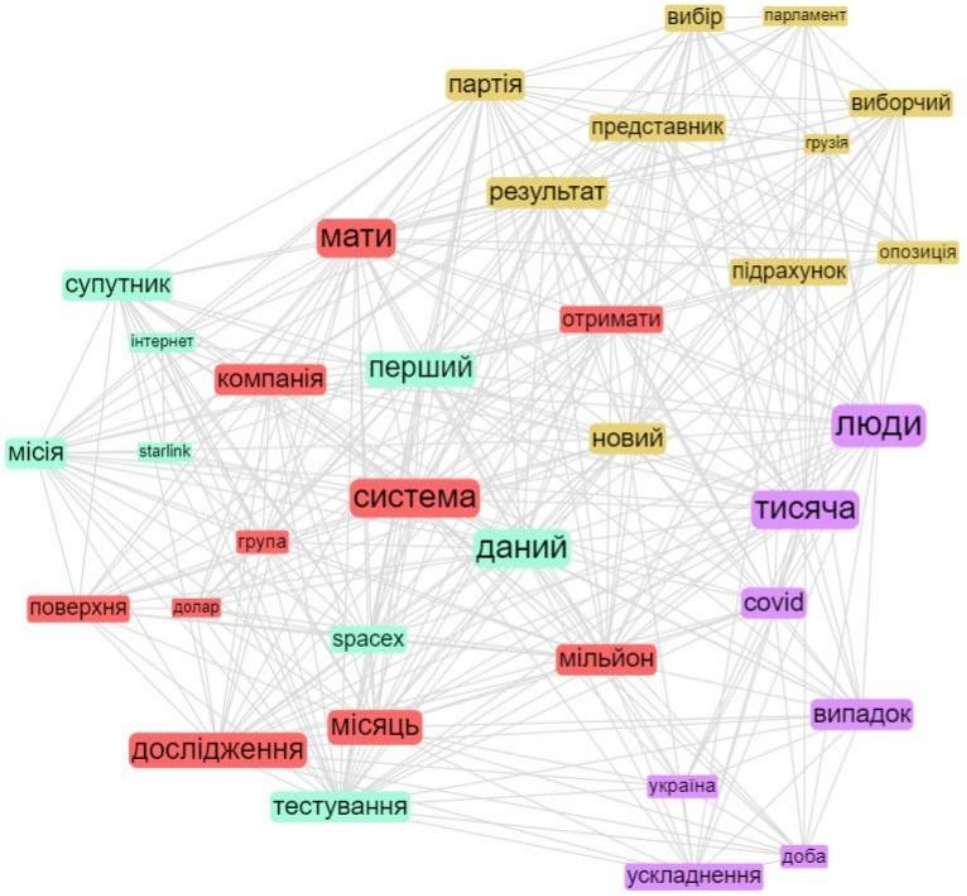


Рисунок 3.7 - Граф термів проаналізованого корпусу текстів

При натисканні на терм на тематичному табло, чи на картці документу, чи в графі (рисунок 3.8), відкривається нове табло, одразу під таблом з темами (рисунок 3.9). Воно містить інформацію про той терм, який користувач виділив кліком.

Рисунок 3.8 - Місця, де терм є інтерактивним елементом

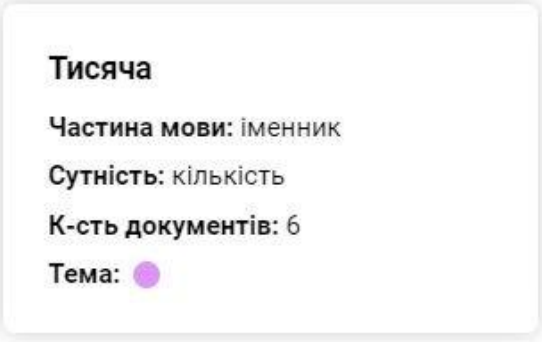


Рисунок 3.9 - Табло з інформацією про виділений терм

На самому табло відображається значення терма, його частина мови, наприклад іменник, його сутність, якщо терм є іменником, кількість документів, а також колір теми до якої він належить.

Окрім нового табла, на графі виділяються усі зв’язки виділеного терма (рисунок 3.10). Завдяки цьому можна зрозуміти з якими іншими термами він зустрічається в документах.

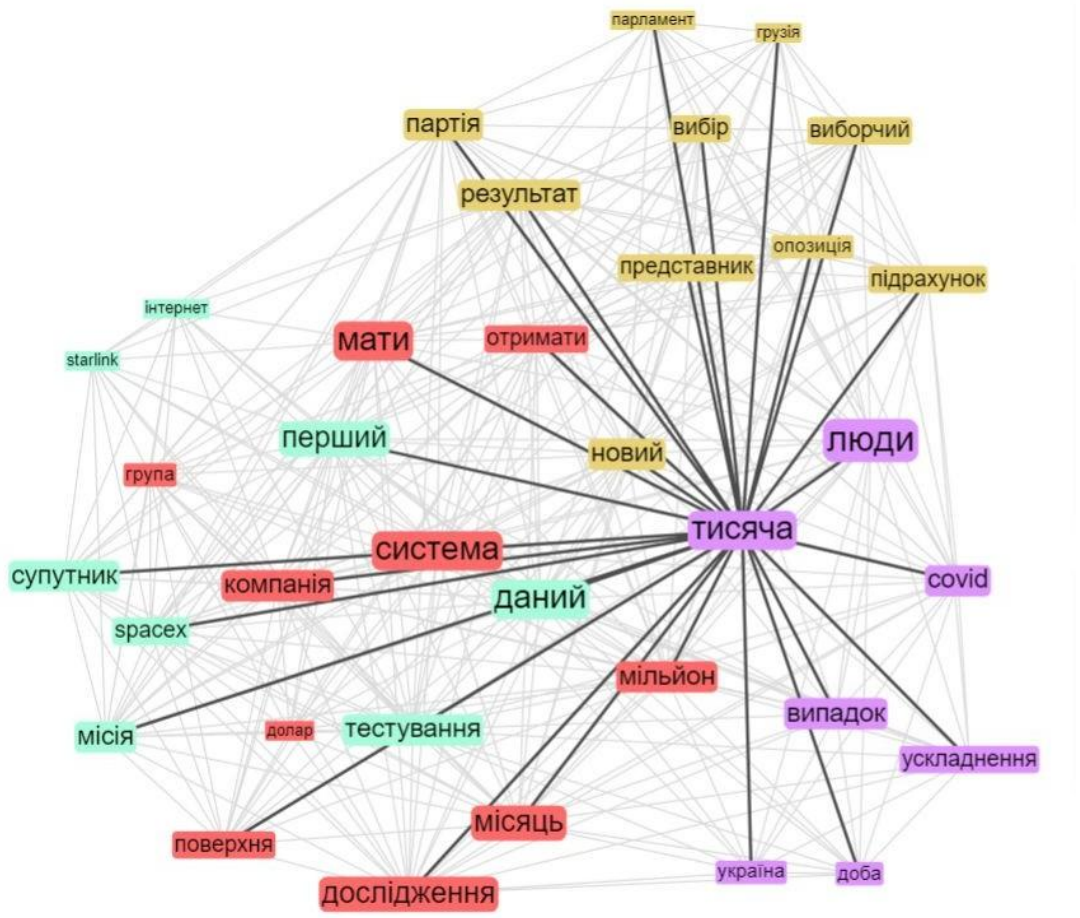


Рисунок 3.10 - Виділення зв’язків обраного терму в графі

Також, в списку документів виділяються жовтим кольором ті, що містять в своєму тексті даний терм (рисунок 3.11).

Рисунок 3.11 - Виділення документів що містять обраний терм

Крім того, при натисканні на ребро графу, він виділяється, а також виділяються усі документи які містять терми, що поєднує даний зв’язок (рисунок 3.12). Завдяки цьому можна легко зрозуміти в якому контексті поєднані дані слова.

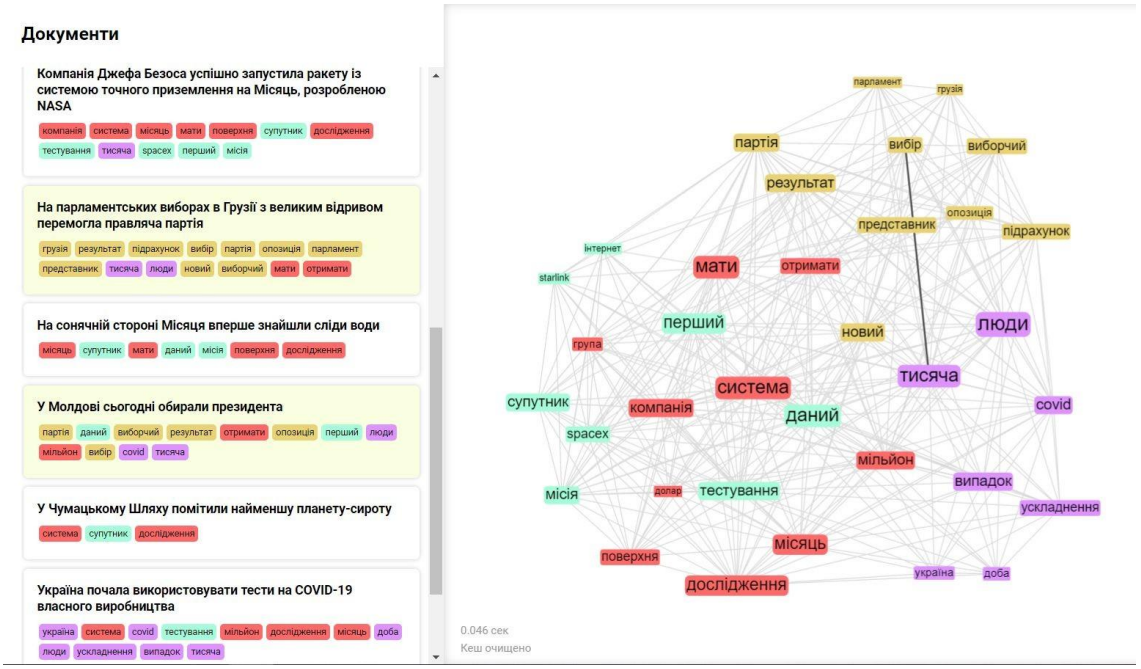


Рисунок 3.12 - Виділення документів, що поєднують обрані терми

Внизу справа від бічної панелі документів є блок, що відображає час, який витратила модель на аналіз даного корпусу текстів з заданими налаштуваннями користувача (рисунок 3.13). Значення відображається в секундах.

Рисунок 3.13 – Блок, що відображає швидкість роботи моделі

Під значенням швидкості розміщена кнопка, при натисканні на яку, користувач може скинути кеш в моделі для даного корпусу, якщо до нього були додані нові документи, або були здійснені зміни в тексті хоча б одного документу. Якщо кеш не буде очищений, то модель не зверне уваги на ці зміни. Після натискання користувач отримує сповіщення про те що кеш було очищено (рисунок 3.14).

Рисунок 3.14 - Сповіщення про очищений кеш моделі

3.3 Опис системи

Оскільки програмне забезпечення являє собою клієнт-серверний веб-застосунок, то для презентації процесу буде зручно показати його у вигляді діаграми діяльності (рисунок 3.15).

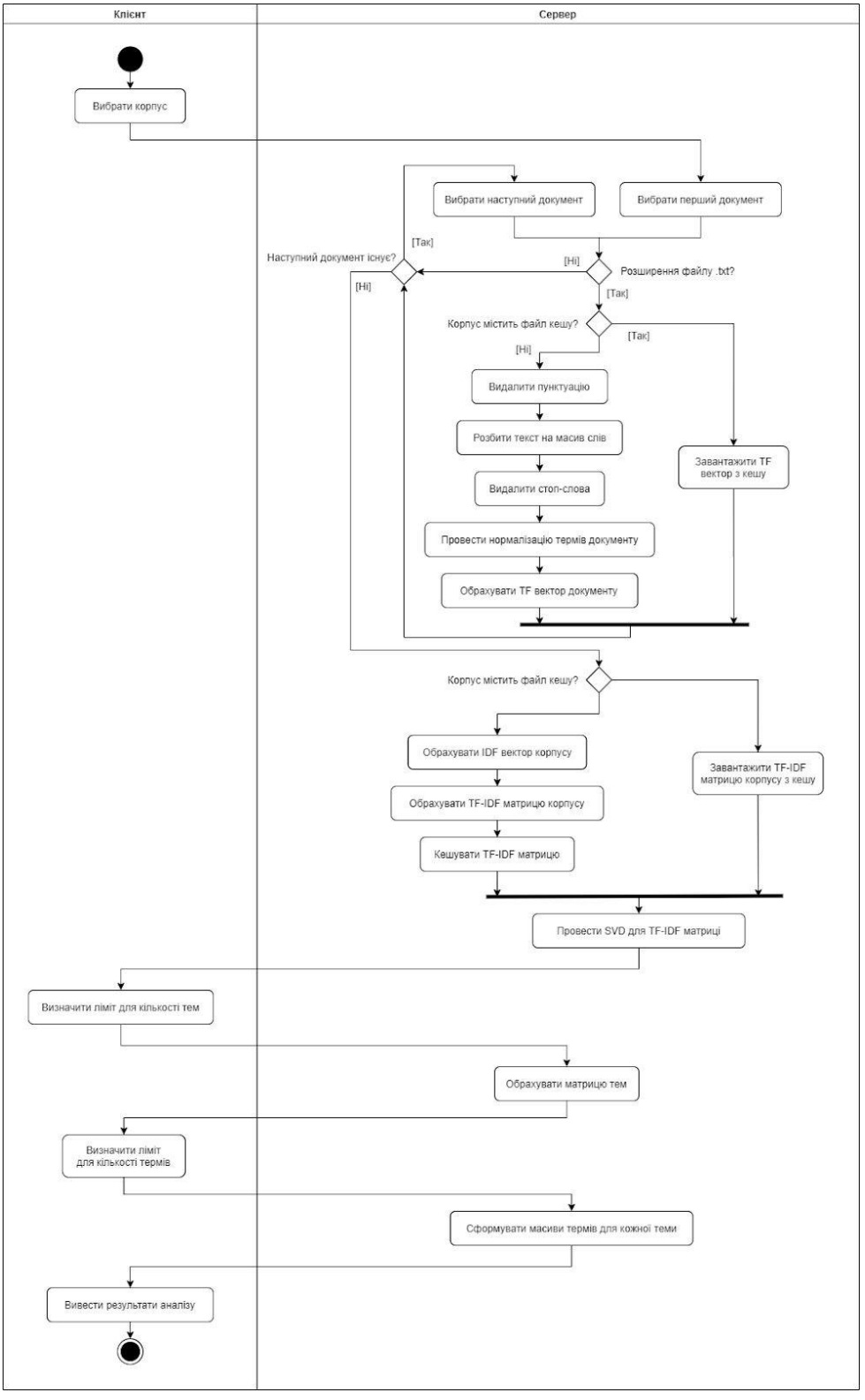


Рисунок 3.15 - Діаграма діяльності розробленої системи

Розглянемо структуру проекту. На рисунку 3.16 зображений список усіх файлів в проекті. Розроблений продукт має модульну архітектуру, адже backend частина працює як REST API, що надає дані для frontend частини.

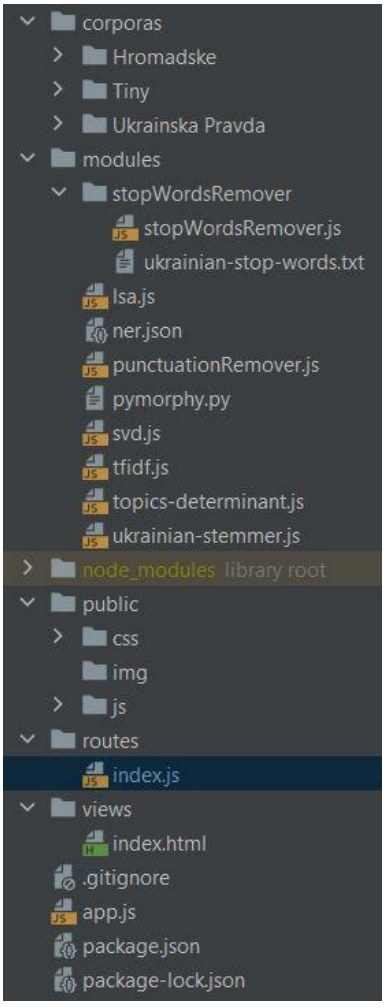


Рисунок 3.16 - Список файлів проекту

Файл routes/index.js визначає за якими запитами клієнт може отримати необхідні для нього дані, а також дані, що має передати клієнт. В папці modules зберігаються модулі, що реалізують певний функціонал, зображений на діаграмі діяльності. В основному всі модулі підключаються до файлу routes/index.js, або один в одного. Завдяки цьому руйнується монолітність застосунку, адже якщо побудується новий модуль, який буде реалізовувати новий метод семантичного аналізу (або інший функціонал) і повертатиме той самий тип даних, то можна з легкістю замінити уже існуючий.

В папці corporas є папки, які називаються відповідно до назви корпусу текстів. Очевидно, що вони містять файли з розширенням .txt, що містять контент документів даного корпусу. Також, у цих папках формуються та зберігаються json файли з кешованими даними отриманими після проведеного аналізу.

Папки public та views містять файли для клієнтської частини застосунку, що описують функціонал та стилі візуального графічного інтерфейсу системи.

В таблиці 3.1 наведені функції, що розроблені на серверній частині проекту в порядку ініціалізації після GET запиту на url “/corporaData” з вхідними даними, а саме назвою корпусу та обмежувачами.

Таблиця 3.1 - Функції серверної частини застосунку

Назва функції	**Вхідні параметри**	**Опис**
loadCorpora	name: string	Функція шукає папку корпусу за значенням параметру name в папці corporas, аналізує усі документи та повертає структуру, що містить назви та контент усіх документів, а також вектор нормалізованих слів, що містить цей документ. Також повертається вектор bag of words, що містить усі унікальні терми знайдені в даному корпусі текстів.

normalizeTerms	termsArray: array	Функція реалізує запит до модулю що написаний мовою python та передає туди масив термів termsArray перетворений у формат JSON. Після отримання результату з модулю pymorphy.py функція повертає масив нормалізованих термів також у JSON форматі.

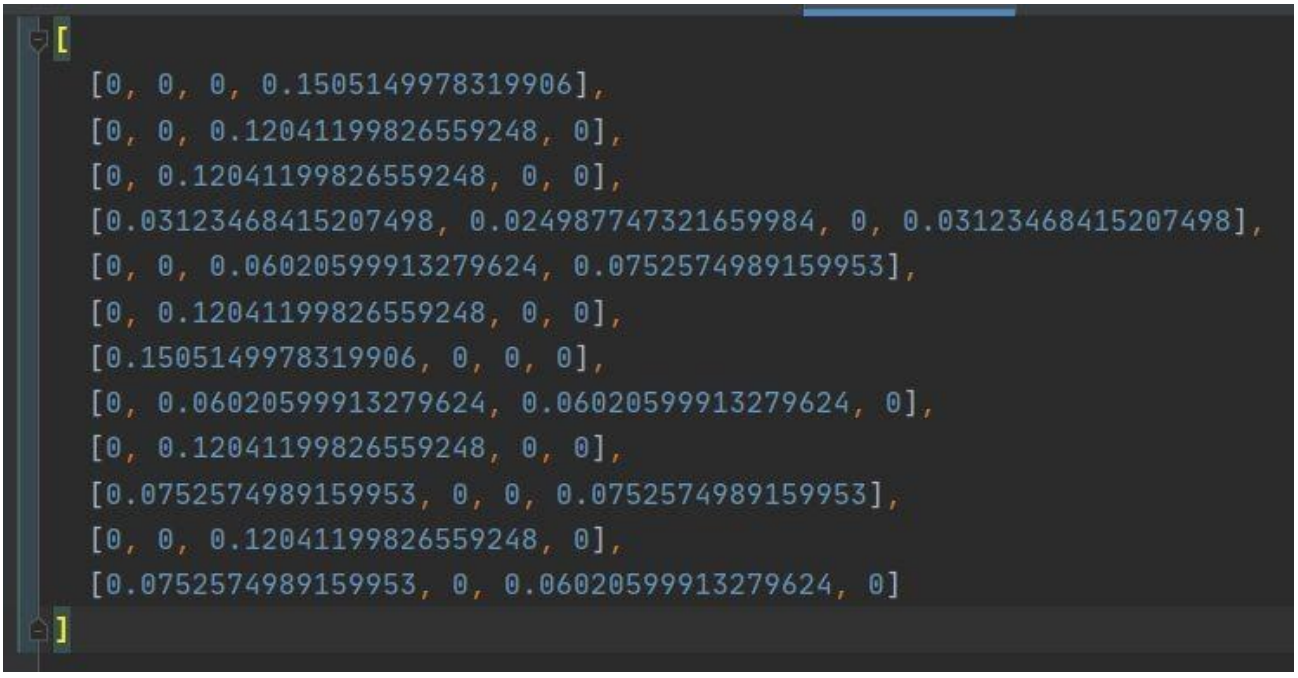
Продовження таблиці 3.1

determineTopics	corpora: structure, topicsLimit: int, termsLimit: int	Функція отримує на вхід структуру, яку повертає функція loadCorpora, а також обмеження що були отримані з запиту клієнта. На виході отримується масив тем, кожна з яких це масив термів, який був описаний в другому розділі.
tfidf	corpora: structure	Функція отримує на вхід структуру, яку повертає функція loadCorpora. На виході отримується TF-IDF матриця, що була описана в другому розділі.
getTopicsMatrix	corporaTFIDF: array, topicsLimit: int	Функція отримує на вхід масив, що повертає функція tdidf, а також обмеження кількості тем, що було отримане з запиту клієнта. На виході отримується матриця тем, що була описана в другому розділі.
svd	tfidf: array	Функція отримує на вхід масив, що повертає функція tdidf. Функція реалізовує метод Singular Value Decomposition, що був описаний в першому розділі. Функція повертає три масиви, що є компонентами декомпозованої TDIDF матриці.

Продовження таблиці 3.1

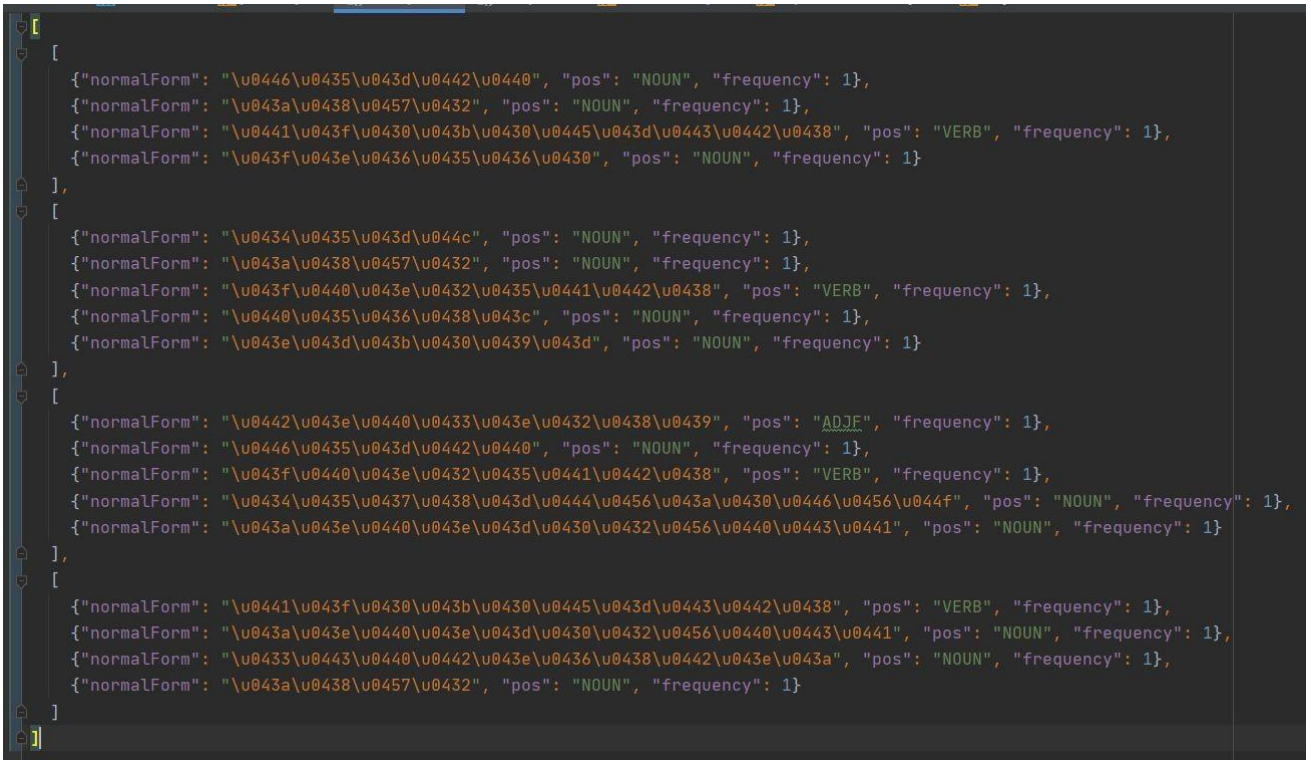
multMatrix	A: array, B: array	Функція на вхід отримує два двомірні масиви і реалізовує перемноження цих матриць і повертає двомірний масив, що є результатом множення.
transponateMatrix	A: array	Функція отримує на вхід двомірний масив, виконує траспонування матриці і повертає двомірний масив, що є результатом транспонування.
diagMatrix	vec: array	Функція отримує на вхід масив, що є вектором, перетворює його на діагональну матрицю з значеннями цього вектору на головній діагоналі і повертає результат.
formatDataForView	corpora: structure, topics: array	Функція отримує на вхід структуру, яку повертає функція loadCorpora, а також матрицю тем, що повертає функція getTopicsMatrix і форматує ці дані в JSON формат, що буде відправлений на клієнт.

Як вже згадувалось раніше система кешує дані, що були отримані в процесі аналізу корпусу текстів, а саме двомірний масив, що являє собою TF-IDF матрицю (рисунок 3.17) та вектори нормалізованих термів для усіх документів корпусу (рисунок 3.18).



```
[
  [0, 0, 0, 0.1505149978319906],
  [0, 0, 0.12041199826559248, 0],
  [0, 0.12041199826559248, 0, 0],
  [0.03123468415207498, 0.024987747321659984, 0, 0.03123468415207498],
  [0, 0, 0.06020599913279624, 0.0752574989159953],
  [0, 0.12041199826559248, 0, 0],
  [0.1505149978319906, 0, 0, 0],
  [0, 0.06020599913279624, 0.06020599913279624, 0],
  [0, 0.12041199826559248, 0, 0],
  [0.0752574989159953, 0, 0, 0.0752574989159953],
  [0, 0, 0.12041199826559248, 0],
  [0.0752574989159953, 0, 0.06020599913279624, 0]
]
```


Рисунок 3.17 - Приклад TF-IDF матриці, завантаженої до кешу



```
[
  [
    {"normalForm": "\u0446\u0435\u043d\u0442\u0440", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043a\u0438\u0457\u0432", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u0441\u043f\u0430\u043b\u0430\u0445\u043d\u0443\u0442\u0438", "pos": "VERB", "frequency": 1},
    {"normalForm": "\u043f\u043e\u0436\u0435\u0436\u0430", "pos": "NOUN", "frequency": 1}
  ],
  [
    {"normalForm": "\u0434\u0435\u043d\u044c", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043a\u0438\u0457\u0432", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043f\u0440\u043e\u0432\u0435\u0441\u0442\u0438", "pos": "VERB", "frequency": 1},
    {"normalForm": "\u0440\u0435\u0436\u0438\u043c", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043e\u043d\u043b\u0430\u0439\u043d", "pos": "NOUN", "frequency": 1}
  ],
  [
    {"normalForm": "\u0442\u043e\u0440\u0433\u043e\u0432\u0438\u0439", "pos": "ADJF", "frequency": 1},
    {"normalForm": "\u0446\u0435\u043d\u0442\u0440", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043f\u0440\u043e\u0432\u0435\u0441\u0442\u0438", "pos": "VERB", "frequency": 1},
    {"normalForm": "\u0434\u0435\u0437\u0438\u043d\u0444\u0456\u043a\u0430\u0446\u0456\u044f", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043a\u043e\u0440\u043e\u043d\u0430\u0432\u0456\u0440\u0443\u0441", "pos": "NOUN", "frequency": 1}
  ],
  [
    {"normalForm": "\u0441\u043f\u0430\u043b\u0430\u0445\u043d\u0443\u0442\u0438", "pos": "VERB", "frequency": 1},
    {"normalForm": "\u043a\u043e\u0440\u043e\u043d\u0430\u0432\u0456\u0440\u0443\u0441", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u0433\u0443\u0440\u0442\u043e\u0436\u0438\u0442\u043e\u043a", "pos": "NOUN", "frequency": 1},
    {"normalForm": "\u043a\u0438\u0457\u0432", "pos": "NOUN", "frequency": 1}
  ]
]
```

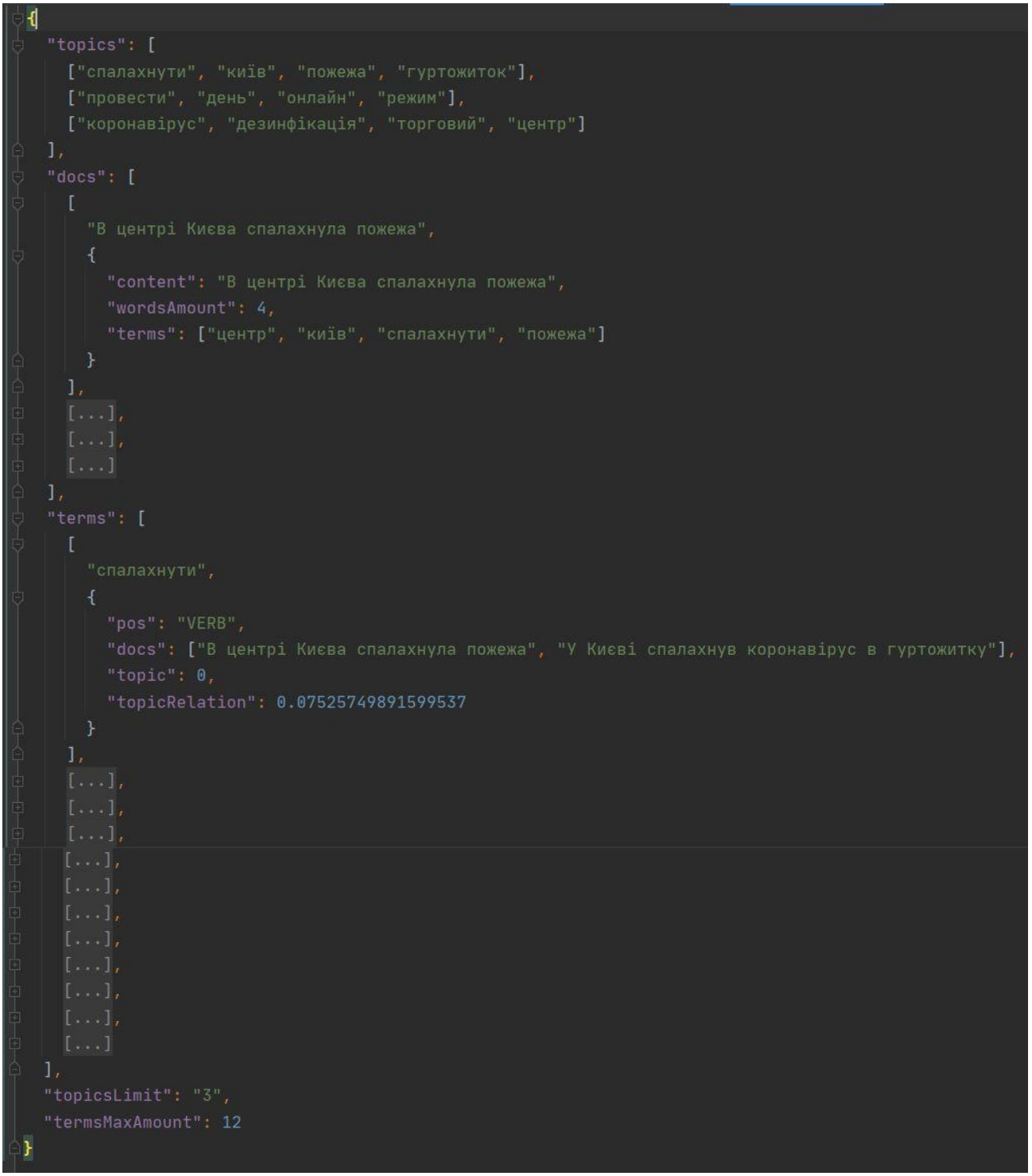

Рисунок 3.18 - Приклад масиву нормалізованих термів, завантаженого до кешу

Розглянемо структуру даних у форматі JSON, яку передає серверна сторона застосунку для клієнтської сторони. Цей об’єкт містить три ключі:

- topics, що є масивом тем, які являють собою масиви термів;

- docs, що є масивом об’єктів з інформацією про документи;
- terms, що є масивом об’єктів з інформацією про терми в даному корпусі.

На рисунку 3.19 зображений приклад даної структури для корпусу текстів, який був обраний як приклад у другому розділі.



```
{
  "topics": [
    ["спалахнути", "київ", "пожежа", "гуртожиток"],
    ["провести", "день", "онлайн", "режим"],
    ["коронавірус", "дезинфікація", "торговий", "центр"]
  ],
  "docs": [
    [
      "В центрі Києва спалахнула пожежа",
      {
        "content": "В центрі Києва спалахнула пожежа",
        "wordsAmount": 4,
        "terms": ["центр", "київ", "спалахнути", "пожежа"]
      }
    ],
    [...],
    [...],
    [...]
  ],
  "terms": [
    [
      "спалахнути",
      {
        "pos": "VERB",
        "docs": ["В центрі Києва спалахнула пожежа", "У Києві спалахнув коронавірус в гуртожитку"],
        "topic": 0,
        "topicRelation": 0.07525749891599537
      }
    ],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...],
    [...]
  ],
  "topicsLimit": "3",
  "termsMaxAmount": 12
}
```


Рисунок 3.19 - Приклад структури що надсилається з сервера на клієнт

В таблиці 3.2 наведені функції, що розроблені на клієнтській частині проекту в порядку виконання після того, як користувач задав вхідні параметри.

Таблиця 3.2 - Функції клієнтської частини застосунку

Назва функції	**Вхідні параметри**	**Опис**
onPageInit	-	Головна функція, що запускається при завантаженні сторінки. Саме вона ініціалізує всі подальші функції, які мають відбутись в той момент коли було побудоване DOM дерево сторінки.
getCorporasList	-	Функція що надсилає запит на серверну частину застосунку, та повертає список існуючих корпусів текстів, що вже завантажені в систему, тобто присутні в папці corporas.
initCorporasSelector	corporasList: array	Функція що ініціалізує селектор, що дозволяє користувачу обрати який корпус буде проаналізований, базуючись на масиву даних, що передаються в якості параметра corporasList.

Продовження таблиці 3.2

showCorpora	name: string	Функція що отримує на вхід ім’я корпусу, який необхідно проаналізувати, ініціалізує функцію loadData, логуючи час за який була отримала відповідь і передає це значення в функцію renderPage.
loadData	name: string	Функція що отримує на вхід ім’я корпусу, який необхідно проаналізувати. Потім вона отримує значення слайдерів, що налаштовують вхідні дані для моделі, та надсилає запит на серверну частину застосунку. Потім ініціалізується метод parseData.
parseData	data: structure	Функція, що аналізує отримані дані з параметру data, які мають таку структуру, яку передає серверна частина застосунку. Також створюються об’єкти, з якими буде маніпулювати функція renderPage.
renderPage	loadingTime: float	Ця функція будує відображення результату роботи системи у графічний інтерфейс, завдяки маніпуляції з об’єктами, що створені функцією parseData. Параметр loadingTime виводиться в блоці, що відображає швидкість роботи (рисунок

		3.13).

Крім цих функцій система ініціалізує усі обробники подій для візуальних елементів з якими буде взаємодіяти користувач після того, як результати аналізу будуть відображені в графічному інтерфейсі.

При зміні будь-якого елементу в основній панелі налаштувань застосунок заново ініціалізує усі вищеописані функції, крім тих, що описують роботу візуальних елементів.

Висновки до розділу

У даному розділі були детально розглянуті технології, що використовувались при розробці системи семантичного аналізу україномовних текстів. Основна технологія - Node.js і відповідно мова програмування JavaScript. Для поєднання системи з морфоаналізатором pymorphy2 використовувалась також мова програмування Python. Для зберігання і передачі даних використовувався формат даних JSON.

Крім того, було наведене керівництво користувача з описом усіх функцій, які користувач може здійснити користуючись системою. Також була описана робота системи, наведена діаграма діяльності, описані структури даних, а також усі функції, що були реалізовані в системі.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї стартап-проекту

Оскільки програмний продукт, що був описаний в третьому розділі, є доступним засобом семантичного аналізу, то стартап-проект буде використовувати розроблений веб-застосунок, проте при тому ж залишаючись окремим програмним продуктом. Це буде сервіс для аналізу корпусу текстів українською мовою для користувачів з системою монетизації за кількістю і розміром завантажених корпусів до сервісу. Пропонується реалізовувати розроблений сервіс за моделлю SaaS (з англійської Software as a Service - програмне забезпечення як послуга) адже ця модель ідеально підходить для надання клієнтам інструменту розв’язку нескладних задач, таких як семантичний аналіз текстів. Детальний опис ідеї стартап проекту наведений в таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап проекту

Зміст	**Напрямок застосування**	**Вигоди для користувача**
Розробити веб-застосунок, що буде поширюватись за моделлю SaaS та надаватиме користувачам можливість виділити теми, та побудувати граф термів для корпусу україномовних текстів, попередньо завантаживши його в	Семантичний аналіз корпусу україномовних текстів, та відображення результатів в інтерактивному графічному інтерфейсі	Не потрібно імплементувати код власноруч, достатньо звернутись до стороннього сервісу що вже імплементує цю задачу. Автоматичне розбиття корпусу текстів на теми.

систему.		

4.2 Аналіз ринкових можливостей стартап-проекту

Перш ніж починати розробку стартапу, необхідно провести аналіз ринкових можливостей, сформувати потенційні групи користувачів та проаналізувати уже наявні пропозиції на ринку. Також, дуже важливим є пошук існуючих потенційних конкурентів у даній сфері. В таблиці 5.2 можна побачити результати проведеного аналізу для стартапу.

Таблиця 5.2 – Попередня характеристика ринку стартап проекту

Показники стану ринку (найменування)	**Характеристика**
Кількість головних гравців, од	0
Загальний обсяг продаж, грн/ум.од (через витрати на продукти-замінники)	3 000 000
Динаміка ринку (якісна оцінка)	Зростаючий
Наявність обмежень для входу (вказати характер обмежень)	Юридичні обмеження (GDPR)

Специфічні вимоги до стандартизації та сертифікації	Вимоги до програмного забезпечення, а також юридичні вимоги.
Середня норма рентабельності в галузі (або по ринку), %	40 %

В середньому відсоток на вкладення в українських банках складає 10% [22] у гривні, а середня норма рентабельності у даній галузі 40%. Саме тому значно вигідніше вкладати гроші в даний проект, зважаючи на відсутність головних гравців, а також кількість прямих конкурентів. Цей ринок є дуже привабливим для потенційних інвесторів, проте необхідно звертати увагу на певні юридичні обмеження, або вимоги до програмного забезпечення. Проте, звертаючи увагу на рентабельність це не буде великою проблемою для інвесторів.

В таблиці 5.3 показані характеристики потенційних клієнтів старптапу. Цільову аудиторію можна розбити на 2 класи - одиничні користувачі та організації, що мають різноманітні набори україномовних текстів.

Таблиця 5.3 – Характеристика потенційних клієнтів застосунку

Потреба, що формує ринок	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**

Велика кількість корпусів текстів українською мовою, що неможливо проаналізувати на іноземних сервісах.	Користувачі, що розробляють, колекціонують чи накопичують україномовні тексти.	Переважно користуються сервісом один чи декілька разів.	Швидкий і якісний аналіз набору україномовних документів.
	Організації, що розробляють, колекціонують чи накопичують україномовні тексти.	Користуються сервісом на регулярній основі.	Надійний та безперебійний доступ до сервісу на регулярній основі.

Зважаючи на те, що клієнтами можуть бути як звичайні користувачі, так і організації, то є сенс розробляти застосунок за двома типами доступу - для B2C та B2B клієнтів. Для цього необхідно буде чітко розділити маркетингову політику, для того щоб захоплювати правильних користувачів в необхідний для них тип.

Потрібно ще розглянути моделі конкурентів за допомогою порівняльної характеристики, наданої за методом п’яти сил М. Портера.

Таблиця 5.4 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**
	Відсутні	Існують рішення для інших мов, але не існує	Хостинг серверної частини

		для української.	
Висновки	Не інтенсивна	Можливо є нішеві конкуренти, що використовують подібні методи. Можливо є потенційні конкуренти у суміжних галузях.	Так. Погодження умов співпраці.

Продовження таблиці 5.4

Складові аналізу	**Клієнти**	**Товари-замінники**
	Кількість користувачів та їх зацікавленість у платній версії.	Часткова заміна, вища якість, підтримка інших мов.
Висновки	Так, якість застосунку, зацікавленість в користуванні.	Так, велика вірогідність що деякі клієнти можуть перекладати тексти та використовувати сервіси для

		іноземних мов.

З даної таблиці можна зробити висновок, що зважаючи на ситуацію з конкурентами на такому ринку можна спокійно працювати. Проте щоб залишатись на ринку конкурентоспроможним, проект має розвивати такі характеристики-переваги:

- висока якість наданих рішень;
- висока швидкість проведення обчислень на великих корпусах текстів;
- нижча ціна за користування сервісом ніж у конкурентів;
- таргетована та клієнтоорієнтована рекламна кампанія.

Останній етап ринкового аналізу це розробка SWOT аналізу – таблиця розміром 2х2 що описує сильні та слабкі сторони стартапу, а також загрози і потенційні можливості. Ринкові загрози та можливості формуються на основі результатів аналізу факторів загроз та можливостей маркетингово середовища. Вони є прогнозованими результатами і є наслідками впливу певних факторів, тому вони ще не реалізовані на ринку але мають певну ймовірність здійснення. Для прикладу фактор зменшення доходів потенційних клієнтів є загрозливим. На його основі можна прогнозувати посилення значущості цінового фактору, що впливатиме на вибір клієнта при виборі сервісу. Також посилиться цінова конкуренція, що є вже ринковою загрозою.

Таблиця 5.5 – SWOT аналіз стартап проекту

Сильні сторони	**Слабкі сторони**

1. Велика кількість корпусів текстів українською мовою, що неможливо проаналізувати на іноземних сервісах. 2. Підвищення актуальності семантичного аналізу текстів. 3. Відсутність конкурентів, що працюють в даній сфері за моделлю SaaS.	1. Розмежування сервісу на B2C та B2B потребує значних витрат на розробку і маркетинг. 2. Високий ризик того, що проект не окупиться.
Можливості	**Загрози**
1. Зменшення операційних витрат у випадку зниження цін на хостинги. 2. Велика кількість клієнтів з України з різних сфер.	1. Обмежений розмір цільової аудиторії. 2. Неготовність українських клієнтів платити за користування сервісом. 3. Утворення ціни за якої буде більш вигідно мати особисте вбудоване рішення.

Підбивши підсумки по SWOT аналізу можна перейти до розробки маркетингової стратегії для стартап продукту.

4.3 Розроблення маркетингової стратегії для стартап продукту

Перший етап маркетингової стратегії це розробка маркетингової концепції товару. В таблиці 5.6 наведені відомості про цю концепцію.

Таблиця 5.6 – Визначення ключових переваг продукту

Потреба	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)**
Семантичний аналіз корпусу україномовних текстів.	Швидкий аналіз завантаженого корпусу текстів і відображення його результатів в інтерактивному графічному інтерфейсі.	Удосконалений та швидший метод тематичного моделювання в розробленій моделі.
Економія часу та грошових ресурсів на розробку.	Економія витрат часу і ресурсів на власну розробку системи для семантичного аналізу україномовних текстів, або переклад документів на іншу мову задля подальшого аналізу.	Сервіс доступний після недовгої процедури реєстрації і не потрібно проводити розробку з нуля та підтримувати його в подальшому.

Другим етапом маркетингової стратегії є ціноутворення. Необхідно відзначити те, що ціноутворення переважно визначають лише під час фінансово-економічного аналізу стартапу. Тому можна опустити це і вирахувати тільки загальну ціну, спираючись на експертну думку. Ціна для одного клієнта за один запит повинна бути вирахувана так, щоб для певної середньої кількості запитів за місяць її добуток з ціною за ліцензію був меншим, ніж вартість розробки свого особистого сервісу включно з операційними коштами. Це вираховується для того, щоб зрозуміти вигідну для користувача, а також і для розробника ціну.

Третій етап це пошук каналів продажу для подальшого збільшення бази користувачів. Зважаючи на те, що буде використовуватись дві різні моделі - B2C та B2B, то необхідно розділити продажі на 2 канали. Для B2B моделі основний канал продажів це зустрічі та презентації сервісу для кожного з потенційних клієнтів.

Також для цього можна розробити спеціальний застосунок для їх подальшого управління. Для моделі B2C найкраще підходить таргетована та вбудована реклама в соцмережах, на сайтах, що поширюють інформацію в даній сфері діяльності а також блогах. Також можна розробити спеціальну реферальну програму, за якою клієнти можуть запрошувати в сервіс своїх друзів чи знайомих та отримувати за це знижку.

Потрібно також надати концепцію маркетингової комунікації для відділу продаж та маркетологів. Основна орієнтація даної концепції полягатиме в фокусі на сучасний підхід SaaS для роботи сервісу, а основною задачею буде переконання потенційного користувача у рентабельності стартапу у порівнянні з іншими сервісами чи підходами.

Висновки до розділу

У даному розділі був проведений аналіз стартапу, що потенційно можна розробити базуючись на програмному рішенні описаному в розділі 3. Був проведений SWOT аналіз, описані характеристики потенційних користувачів, аналіз конкуренції за системою п’яти сил М. Портера а також характеристики ринку.

Підводячи підсумок з аналізів можна визначити стартап як нішевий та унікальний сервіс без прямих конкурентів, лише з сервісами-замінниками. Для початку необхідно розробити лише MVP системи і залучені кошти виділяти повністю на маркетинг та відділ продажів. Зважаючи на ринок та на характеристики даного продукту, то розроблюваний стартап має швидко вийти на прибутковість.

5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО МЕТОДУ

5.1 Аналіз швидкодії моделі

Для дослідження ефективності роботи моделі було вирішено робити заміри швидкодії в залежності від кількості документів в корпусі. Така метрика була обрана оскільки недоліком методу латентно-семантичного аналізу є висока обчислювальна складність і низька швидкість роботи, що вимагає повторного обчислення усіх значень для всього корпусу в разі додавання нового документа. Проте завдяки оптимізації збереження обчислень значно збільшилась швидкість роботи системи.

Розглянемо діаграму на рисунку 5.1, де зображений графік залежності швидкості роботи моделі від кількості документів в корпусі україномовних текстів. Швидкість вимірюється в секундах. Лінія синього кольору показує швидкодію моделі з включеним кешуванням обчислень, а червоного кольору - без кешування.

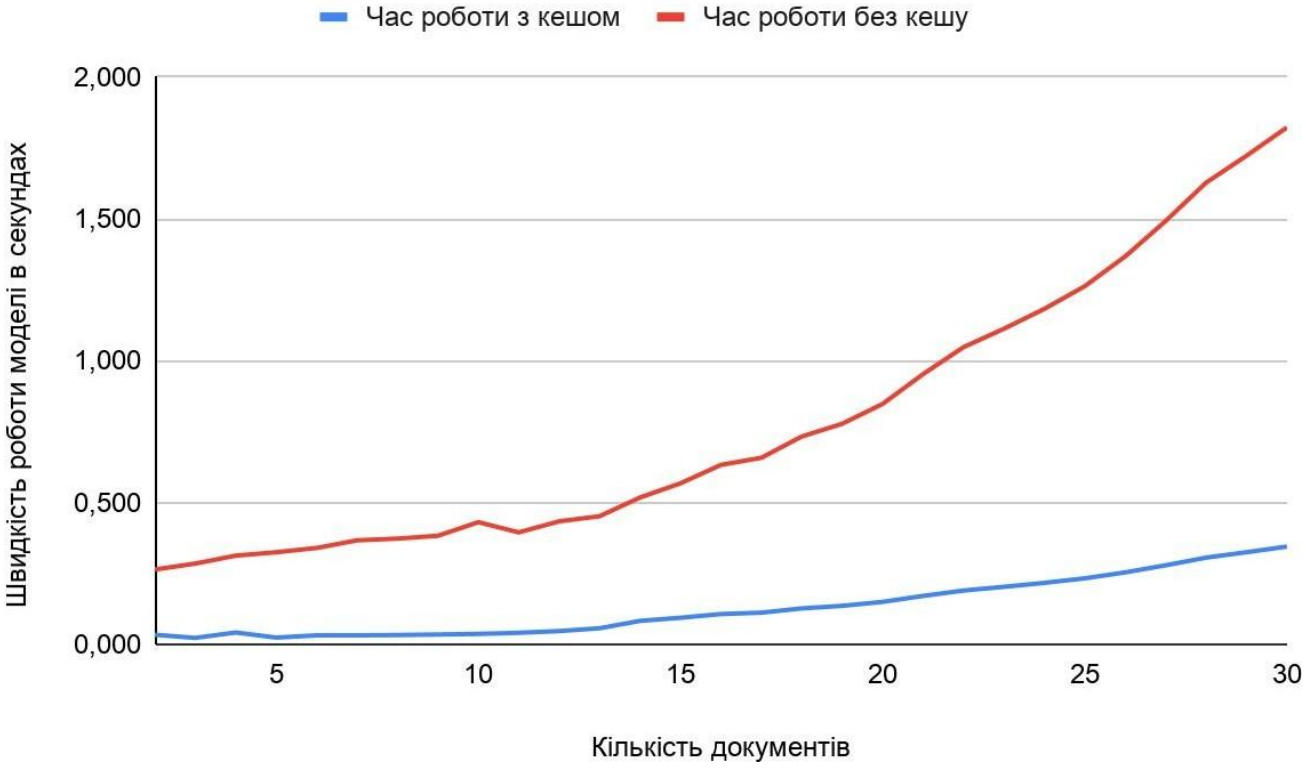


Рисунок 5.1 - Графік залежності швидкості проведення аналізу від кількості документів в корпусі текстів

Щоб зрозуміти наскільки складнішою стає обробка, на рисунку 5.2 наведений графік загальної кількості слів, що обробляє система в залежності від кількості документів в корпусі.

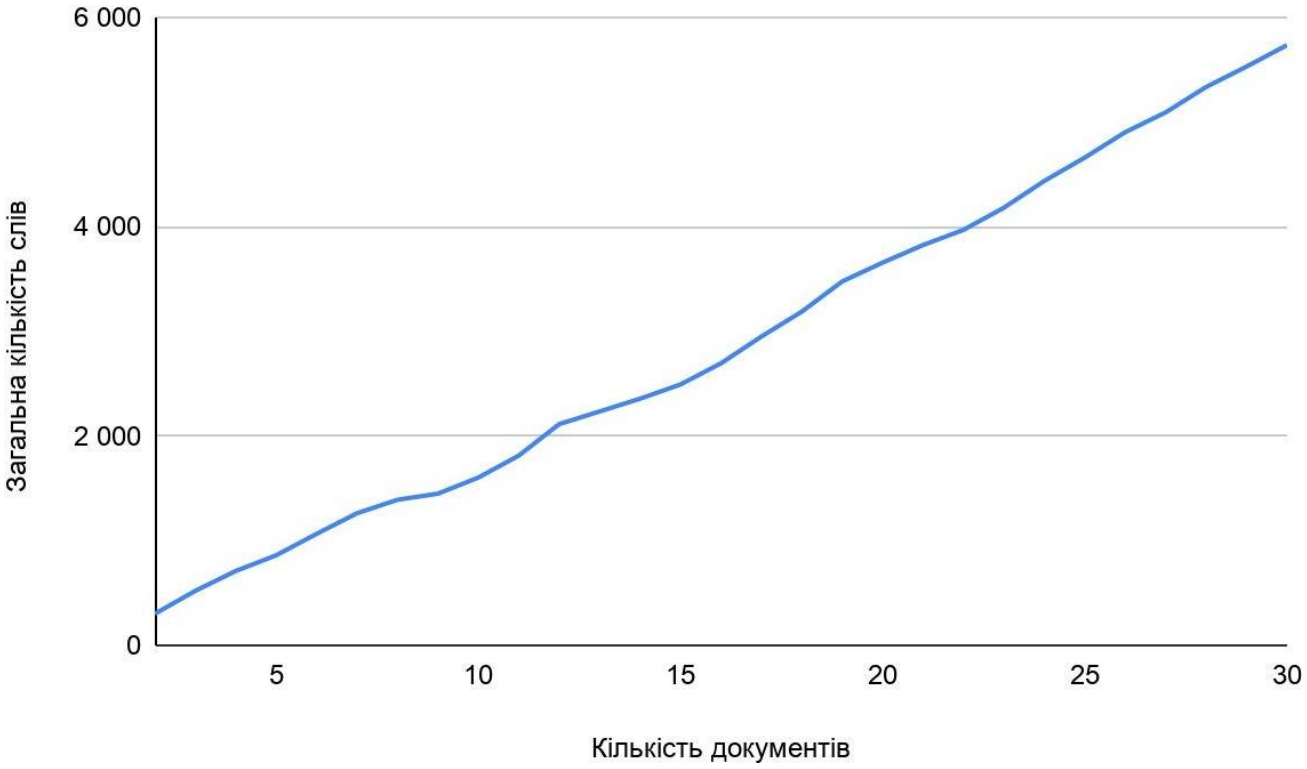


Рисунок 5.2 - Графік залежності загальної кількості слів від кількості документів в корпусі

Як бачимо на рисунку 5.2 кількість слів у документах підібрана майже однаково, для того, щоб рівномірно збільшувати навантаження на систему з кожним наступним документом.

З графіку на рисунку 5.1 можна зрозуміти що розроблений удосконалений метод працює в декілька разів швидше ніж звичайний метод латентно-семантичного аналізу. Зі збільшенням кількості документів в корпусі, різниця між швидкістю роботи моделі з кешем і без нього стає значнішою, але в такій же ж пропорції.

Наприклад, для корпусу що містить лише 2 документи, швидкість роботи моделі без кешу становить 264 мілісекунди, порівняно з швидкістю роботи моделі з кешем - 34 мілісекунди (відношення ~1/7). Якщо взяти корпус

розміром 30 документів, то швидкість роботи моделі без кешу становить 345 мілісекунди, порівняно з швидкістю роботи моделі з кешем - 1824 мілісекунди (відношення ~1/6).

Таким чином можна зробити висновок що завдяки удосконаленню методу вдалось збільшити швидкість роботи системи приблизно в 6 разів, що є доволі значним показником.

5.2 Розподілений режим обробки

Існує ще один спосіб збільшити швидкодію розробленого програмного засобу за допомогою розподіленого режиму обробки вхідних даних та розподілених обчислень. Даний застосунок можна використати як функціонал воркеру у кластері. Єдиний екземпляр Node.js може запускатись лише в одному потоці, і за замовчуванням він має обмеження пам'яті 512 мегабайт в 32-розрядних системах і 1 гігабайт в 64-розрядних системах. Хоча обмеження пам'яті може бути збільшено до близько одного гігабайту в 32-розрядних системах і 1,7 гігабайт в 64-розрядних системах, обсяг пам'яті і обчислювальна потужність можуть стати вузькими місцями для різних процесів.

Елегантне рішення, яке Node.js надає для масштабування додатків, полягає в розподіленні одного процесу на декілька процесів чи воркерів. Це може бути досягнуто через модуль Cluster. Цей модуль дозволяє створювати дочірні процеси (worker), які спільно використовують всі порти сервера з основним процесом Node (master). Теоретично застосування даного підходу дасть лінійний приріст швидкодії в стільки разів, скільки воркерів можливо буде ініціалізувати на обраному сервері.

Кластер - це пул схожих воркерів, які працюють під батьківським процесом Node. Воркери створюються за допомогою методу fork() модуля child_processes. Це означає, що воркерии можуть ділитися дескрипторами сервера і використовувати IPC для зв'язку з батьківським процесом Node.

Батьківський процес відповідає за ініціювання воркерів і контроль над ними. Можна створити будь-яку кількість воркерів в нашому майстер-процесі.

Крім того за замовчуванням вхідні з'єднання розподіляються між воркерами за принципом циклічного перебору, яку документація Node.js пропонує використовувати за замовчуванням в якості політики планування.

Висновки до розділу

У даному розділі було проведене дослідження ефективності розробленого удосконаленого методу семантичного аналізу корпусу україномовних текстів. Дослідження показало, що завдяки оптимізації збереження обчислень значно збільшилась швидкість роботи системи, приблизно у 6 разів. Крім того були розглянуті способи подальшого покращення ефективності програмного засобу з метою підвищення продуктивності його робити.

ВИСНОВКИ

У даній магістерській дисертації були розглянуті декілька поширених засобів семантичного аналізу текстів іноземними мовами, а також україномовних текстів. Засобів для англійської та російської мов значно більше, проте є важливі ресурси, які можна використовувати як допоміжні при розробці моделі семантичного аналізу для української мови. Серед них такі сервіси, як InfraNodus, WordWanderer, WordTree для англійської мови, та Istio для російської. Серед існуючих засобів семантичного аналізу україномовних текстів було розглянуто морфоаналізатор pymorphy2, а також морфосинтаксичний аналізатор. Для реалізації системи обрано використовувати відкриту бібліотеку pymorphy2 з метою удосконалення ефективності. Також було здійснено аналіз та порівняння методів тематичного моделювання корпусу текстів, а саме векторну модель текстів, летантно-семантичний аналіз, імовірнісний латентно-семантичний аналіз та латентне розміщення Діріхле.

Було визначено, що результатом магістерської дисертації повинна стати модель, що проводить тематичне моделювання корпусу україномовних текстів. Модель повинна базуватись на вищезгаданих методах семантичного аналізу адаптованих під українську мову. Для збільшення функціональності модель повинна використовувати доступні існуючі засоби семантичного аналізу для української мови. Для підвищення ефективності необхідно оптимізувати збереження даних, які будуть повторно використовуватися в обчисленнях. Для покращення сприйняття результатів моделювання необхідно розробити візуальний графічний інтерфейс з інтуїтивним та простим відображенням результатів роботи моделі у вигляді графу термів.

Розроблено модифікований метод семантичного аналізу тексту, який дає можливість аналізу україномовного контенту шляхом застосування методу латентно-семантичного аналізу та морфоаналізатора Pymorphy2. Семантичні можливості також було збільшено завдяки використанню NER-моделі. В дисертації було продемонстровано роботу моделі на реальному прикладі з відображенням усіх проміжних кроків та результатів.

Програмну реалізацію системи було виконано за допомогою технології Node.js. Також використовувалась мова python для взаємодії з морфоаналізатором pymorphy2. В дисертації наведена діаграма діяльності, описані структури даних, а також усі функції реалізовані в системі. Також, наведене керівництво користувача, з описом усіх функції які користувач може здійснити користуючись системою.

Крім того, був проведений аналіз стартап проєкту, що потенційно можна розробити базуючись на програмному рішенні описаному в дисертації. Був проведений SWOT аналіз, описані характеристики потенційних користувачів, аналіз конкуренції за системою п’яти сил М. Портера а також характеристики ринку. Можна визначити наш стартап, як нішевий та унікальний сервіс без прямих конкурентів, лише з сервісами-замінниками. Зважаючи на ринок та на характеристики даного продукту, то розроблюваний стартап має швидко вийти на прибутковість.

У підсумку було проведене дослідження ефективності розробленого удосконаленого методу семантичного аналізу корпусу україномовних текстів. Для збільшення швидкодії методу, модель запам’ятовує нормалізовані терми та TF-IDF матриці кожного проаналізованого корпусу. Дослідження показало, що завдяки оптимізації обчислень значно збільшується швидкість роботи системи, приблизно у 6 разів, а швидкість аналізу корпусу з 12 документів скорочується з однієї секунди до 0.1 секунди.

В подальшому розвитку програмного засобу необхідно реалізувати вибір різних алгоритмів для тематичного моделювання, а також проводити детальніший аналіз унікальних термів. Для більшої швидкодії об’ємних корпусів можна запровадити паралельну обробку даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. worldwidewebsize.com [Електронний ресурс] : [Веб-сайт] – Режим доступу: www.worldwidewebsize.com (дата звернення: 28.05.2020)
2. w3techs.com [Електронний ресурс] : [Веб-сайт] – Режим доступу: www.w3techs.com/technologies/overview/content_language (дата звернення: 28.05.2020).
3. Paranyushkin, D (2019). InfraNodus: Generating Insight Using Text Network Analysis, Proceedings of WWW ’19 The World Wide Web Conference, Pages 3584-3589, San Francisco, CA, USA.
4. Dörk, M. and D. Knight. 2015. ‘WordWanderer: a navigational approach to text visualisation’, Corpora 10 (1), pp. 83–94.
5. M. Wattenberg and F. B. Viégas, "The Word Tree, an Interactive Visual Concordance," in IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 6, pp. 1221-1228, Nov.-Dec. 2008, doi: 10.1109/TVCG.2008.172.
6. lang.org.ua [Електронний ресурс] : [Веб-сайт] – Режим доступу: www.lang.org.ua (дата звернення: 28.05.2020).
7. mova.institute [Електронний ресурс] : [Веб-сайт] – Режим доступу: www.mova.institute/аналізатор (дата звернення: 28.05.2020).
8. Глушков, Н. А. Анализ методов тематического моделирования текстов на естественном языке / Н. А. Глушков. — Текст : непосредственный // Молодой ученый. — 2018. — № 19 (205). — С. 101-103. — Режим доступу: https://moluch.ru/archive/205/50247/ (дата звернення: 28.05.2020).
9. lucene_uk [Електронний ресурс] : [Веб-сайт] – Режим доступу: www.github.com/arysin/lucene_uk (дата звернення: 28.05.2020).
10. senyk.poltava.ua [Електронний ресурс] : [Веб-сайт] – Режим доступу: www.senyk.poltava.ua/projects/ukr_stemming/stemming_about.html (дата звернення: 28.05.2020).
11. Information retrieval document search using vector space model in R — 2017. — [Електронний ресурс]. Режим доступу: www.datasciencecentral.com/profiles/blogs/information-retrieval-document-

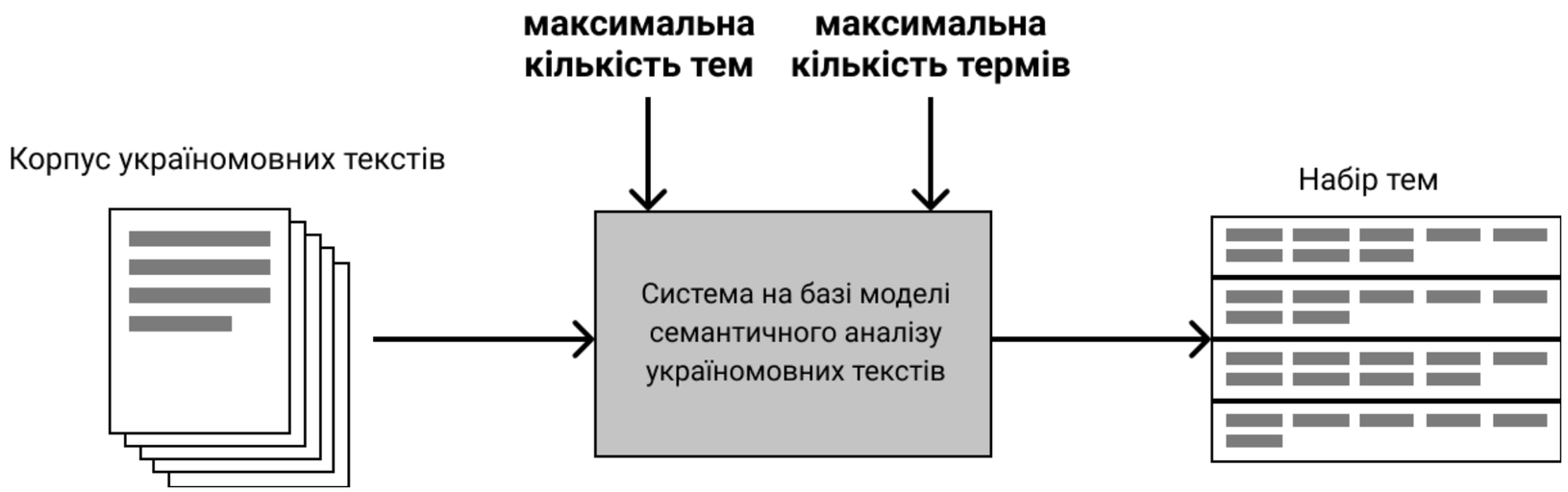
search-usin g-vector-space-model-in (дата звернення: 28.05.2020).

12. Topic Modeling with LSA, PLSA, LDA & lda2Vec — 2018. — [Електронний ресурс]. Режим доступу: www.medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b 05 (дата звернення: 28.05.2020).
13. Topic Modelling with LSA and LDA — 2018. — [Електронний ресурс]. Режим доступу: https://forestforthetree.com/statistics/2018/01/28/topic-modelling-with-lsa-and-lda.html (дата звернення: 23.11.2020).
14. Латентно-семантический анализ — 2010. — [Електронний ресурс]. Режим доступу: https://habr.com/ru/post/110078/ (дата звернення: 20.11.2020).
15. Ю.О. Олійник, О.Є. Афанасьєва, Г.Д.Аршакян Підхід до виявлення аномалій в потоках текстових даних. «Системні технології» 2 (127) 2020 – С.126-139. DOI: https://doi.org/10.34185/1562-9945-2-127-2020-10
16. Мокроусов М.Н. Разработка и исследование методов и системы семантического анализа естественно-языковых текстов — 2010, — Режим доступу: http://tekhnosfera.com/razrabotka-i-issledovanie-metodov-i-sistemy-semanticheskogo-analiza-estestvenno-yazykovyh-tekstov
17. Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. 2002. Topic-based document segmentation with probabilistic latent semantic analysis. In Proceedings of the eleventh international conference on Information and knowledge management (CIKM '02). Association for Computing Machinery, New York, NY, USA, 211–218. DOI:https://doi.org/10.1145/584792.584829
18. K.R. Canini, L. Shi, and T.L. Griffiths. Online inference of topics with latent Dirichlet allocation. In Proceedings of the International Conference on Artificial Intelligence and Statistics, volume 5, 2009.
19. Sandro Pasquali, Mastering Node.js, Packt Publishing Ltd, — 2013, — Режим доступу: https://books.google.com.ua/books?id=fOgtAgAAQBAJ&pg=PA328&dq=node+js&hl=uk&sa=X&ved=2ahUKEwiMo6LB4svtAhWlpYsKHWllDOsQ6AEwCHoECAYQAg#v=onepage&q=node%20js&f=false

20. Hans Petter Langtangen, Python Scripting for Computational Science, Том 3 з серії Texts in Computational Science and Engineering, Springer Science & Business Media, 2009 — Режим доступу: https://books.google.com.ua/books?id=YEoiYr4H2A0C&pg=PA184&dq=python&hl=uk&sa=X&ved=2ahUKEwi5vsXa48vtAhVD_SoKHauzAx4Q6AEwB3oECAgQAg#v=onepage&q=python&f=false
21. Ben Smith, Beginning JSON, Expert's voice in Web development, Apress, 2015 — Режим доступу: https://books.google.com.ua/books?id=ZYYnCgAAQBAJ&pg=PA37&dq=json&hl=uk&sa=X&ved=2ahUKEwis2cyk58vtAhVcCRAIHTqyCfQQ6AEwBHoECAUQAg#v=onepage&q=json&f=false
22. Депозити в банках України [Електронний ресурс] : [Веб-сайт] – Режим доступу: https://minfin.com.ua/ua/deposits/ (Дата звернення: 18.11.2020)

ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ

ДІАГРАМА ПОТОКІВ ДАНИХ У СИСТЕМІ



Демонстраційний плакат до магістерської дисертації

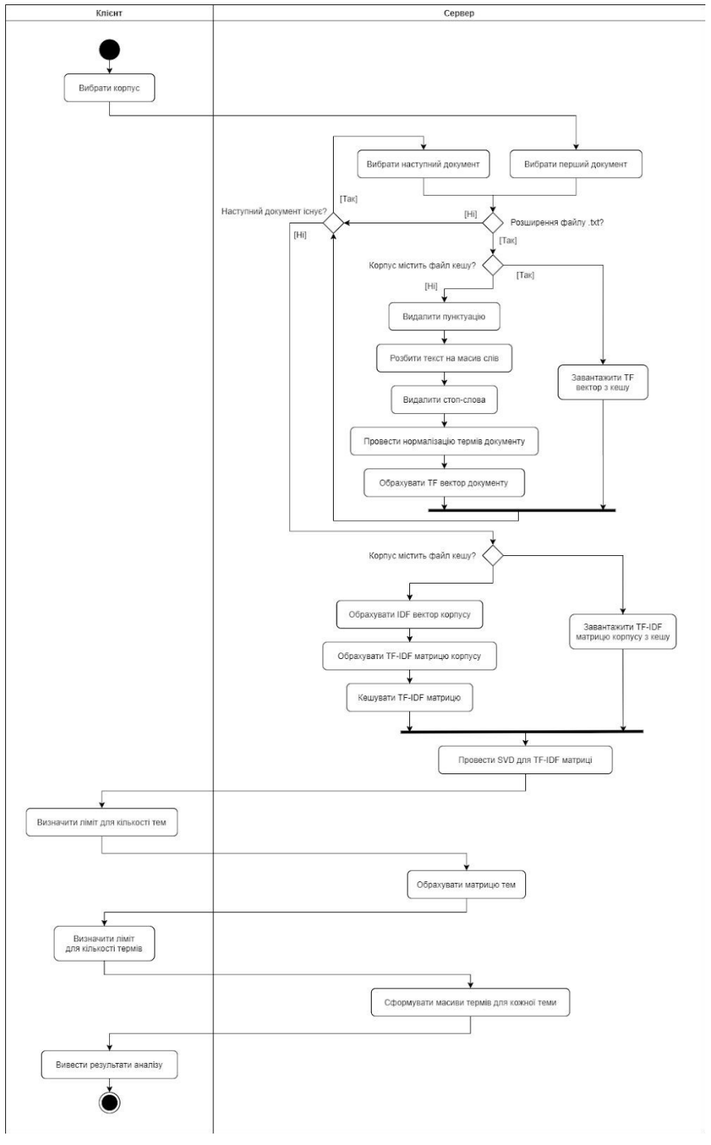
на тему «Методи та засоби семантичного аналізу текстів»

Діаграма потоків даних у системі

Виконав студент гр. ІС-92мп Мигаль Дмитро Степанович

Керівник Олійник Юрій Олександрович

ДІАГРАМА ДІЯЛЬНОСТІ



Демонстраційний плакат до магістерської дисертації

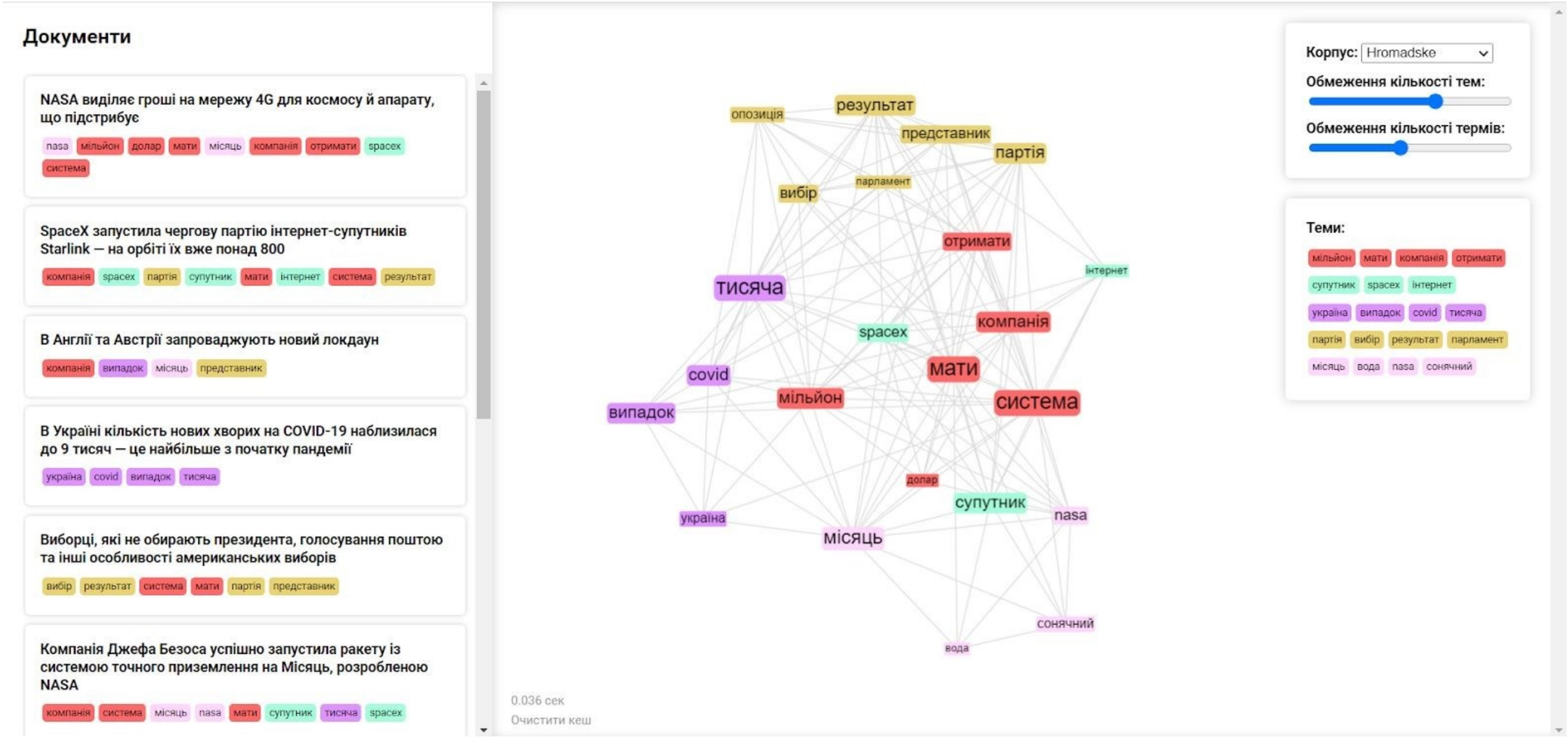
на тему «Методи та засоби семантичного аналізу текстів»

Діаграма діяльності

Виконав студент гр. ІС-92мп Мигаль Дмитро Степанович

Керівник Олійник Юрій Олександрович

КРЕСЛЕННЯ ЕКРАННИХ ФОРМ



Демонстраційний плакат до магістерської дисертації

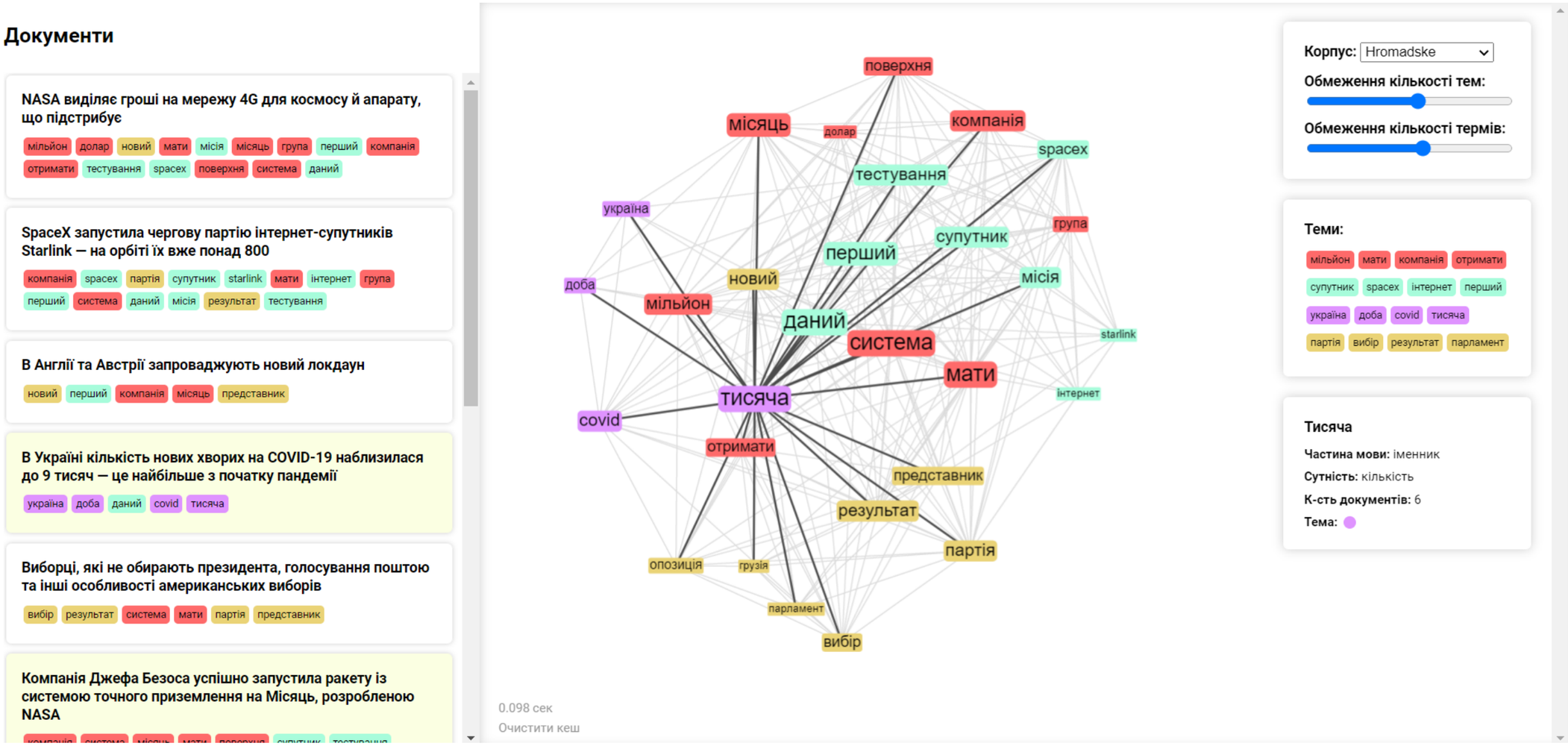
на тему «Методи та засоби семантичного аналізу текстів»

Креслення екранних форм

Виконав студент гр. ІС-92мп — Мигаль Дмитро Степанович

Керівник — Олійник Юрій Олександрович

ВІДОБРАЖЕННЯ РЕЗУЛЬТАТІВ СЕМАНТИЧНОГО ДОСЛІДЖЕННЯ СЛОВА «ТИСЯЧА»



Демонстраційний плакат до магістерської дисертації

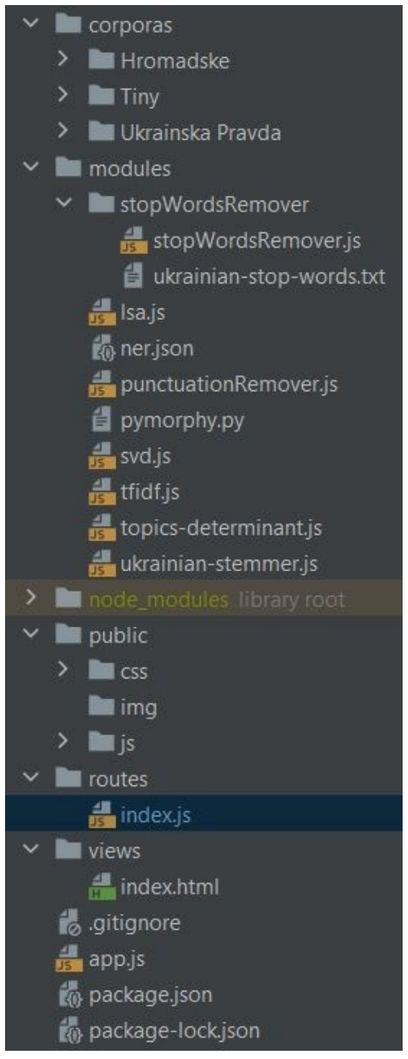
на тему «Методи та засоби семантичного аналізу текстів»

Відображення результатів семантичного дослідження слова «тисяча»

Виконав студент гр. ІС-92мп — Мигаль Дмитро Степанович

Керівник — Олійник Юрій Олександрович

СТРУКТУРА РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАСТОСУНКУ



Демонстраційний плакат до магістерської дисертації

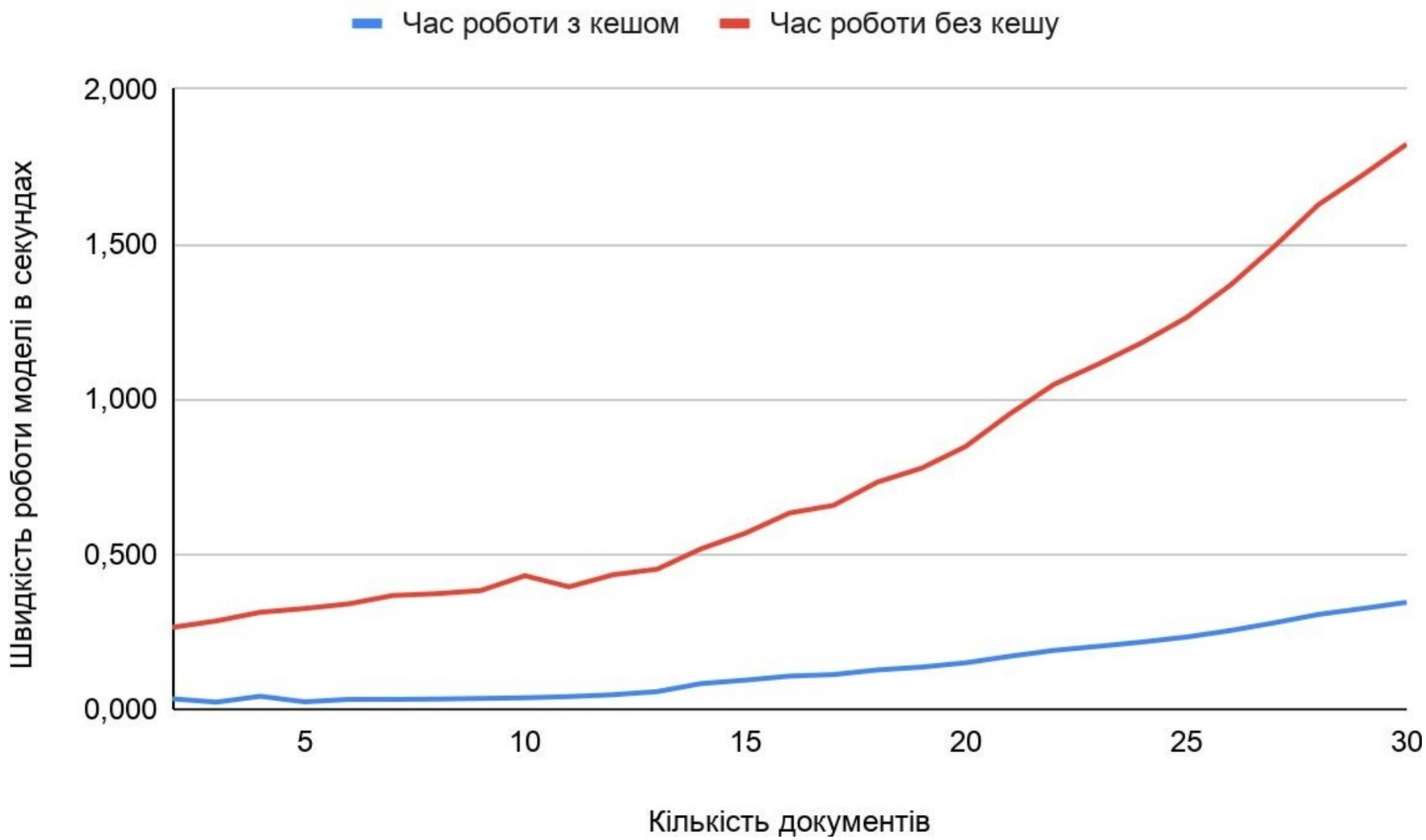
на тему «Методи та засоби семантичного аналізу текстів»

Структура розробленого програмного застосунку

Виконав студент гр. ІС-92мп Мигаль Дмитро Степанович

Керівник Олійник Юрій Олександрович

ЕФЕКТИВНІСТЬ РОБОТИ СИСТЕМИ НА БАЗІ РОЗРОБЛЕНОЇ МОДЕЛІ СЕМАНТИЧНОГО АНАЛІЗУ



Демонстраційний плакат до магістерської дисертації

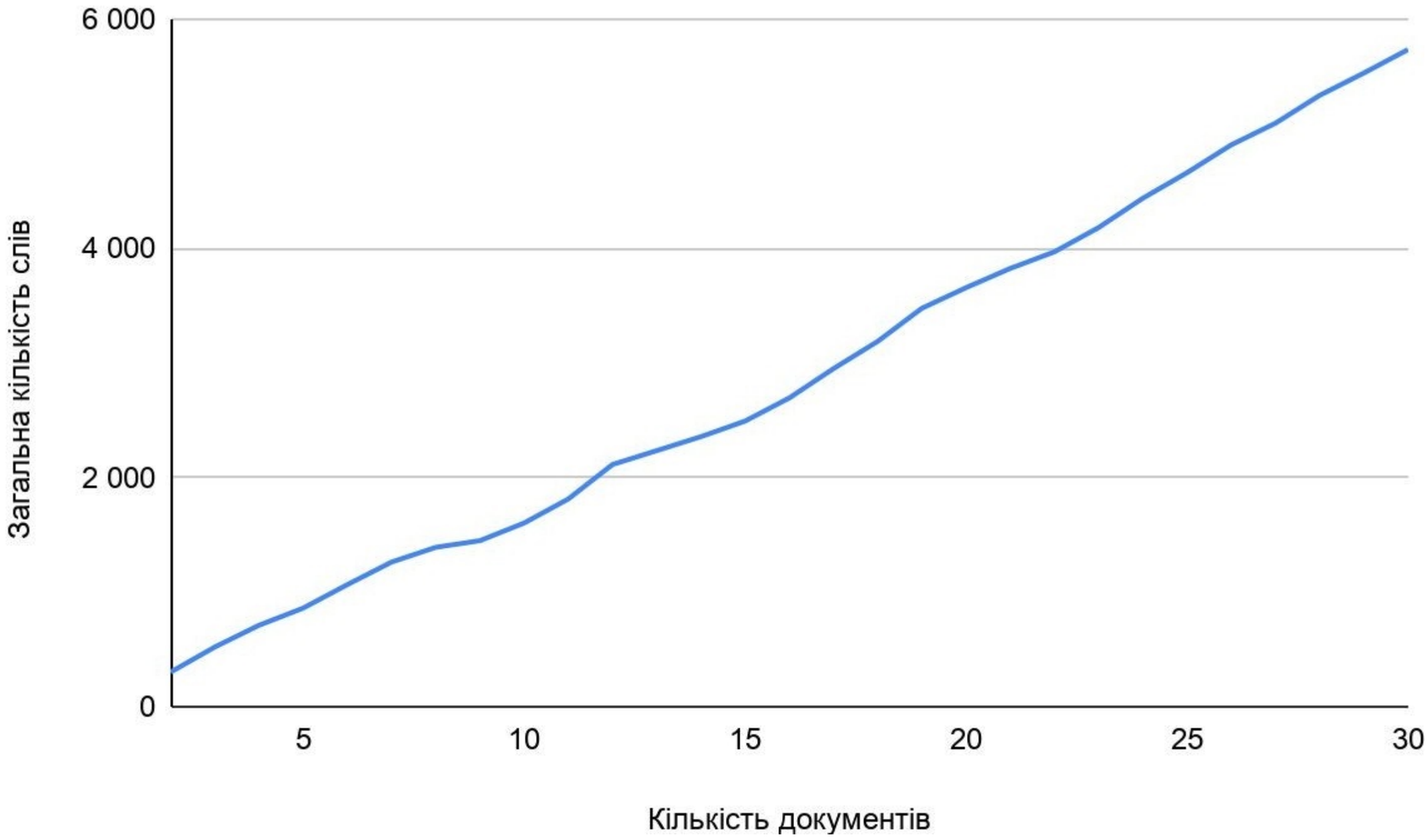
на тему «Методи та засоби семантичного аналізу текстів»

Ефективність роботи системи на базі розробленої моделі семантичного аналізу

Виконав студент гр. ІС-92мп — Мигаль Дмитро Степанович

Керівник — Олійник Юрій Олександрович

ГРАФІК ЗАЛЕЖНОСТІ ЗАГАЛЬНОЇ КІЛЬКОСТІ СЛІВ ВІД КІЛЬКОСТІ ДОКУМЕНТІВ У КОРПУСІ



Демонстраційний плакат до магістерської дисертації

на тему «Методи та засоби семантичного аналізу текстів»

Графік залежності загальної кількості слів від кількості документів у корпусі

Виконав студент гр. ІС-92мп Мигаль Дмитро Степанович

Керівник Олійник Юрій Олександрович